Multi-Factor Authentication on Cloud

Salman H. Khan[1], M. Ali Akbar[2]
[1]The University of Western Australia, Crawley, WA 6009
[2]IOActive, Inc., Seattle WA 98104
Email: salman.khan@uwa.edu.au, muhammad.akbar@ioactive.com

***Abstract*—Due to the recent security infringement incidents of single factor authentication services, there is an inclination towards the use of multi-factor authentication (MFA) mechanisms. These MFA mechanisms should be available to use on modern hand-held computing devices like smart phones due to their big share in computational devices market. Moreover, the high social acceptability and ubiquitous nature has attracted the enterprises to offer their services on modern day hand-held devices. In this regard, the big challenge for these enterprises is to ensure security and privacy of users. To address this issue, we have implemented a verification system that combines human inherence factor (handwritten signature biometrics) with the standard knowledge factor (user specific passwords) to achieve a high level of security. The major computational load of the aforementioned task is shifted on a cloud based application server so that a platform-independent user verification service with ubiquitous access becomes possible. Custom applications are built for both the iOS and Android based devices which are linked with the cloud based two factor authentication (TFA) server. The system is tested on-the-run by a diverse group of users and 98.4% signature verification accuracy is achieved.** [1]

Keywords*—Two-factor authentication, Handwritten signatures, Biometric template, Dynamic signature verification, Distance matching, Hand-held devices.

I. Introduction

With the increasing trend towards ubiquitous computing and Internet technology, remote access to services and private networks is becoming a peculiar feature of today businesses. These advances in technology have facilitated both the enterprises and their targeted user-groups or clients. In a recent report by Gartner, it is estimated that the user authentication services used by enterprises will rise from less than 10% as of today, to more than 50% by 2017 [1]. However, the associated security challenges related to user authenticity and safety of private data have opened new avenues for malevolent activities. The emerging requirement is to provide better security solutions that could efficiently cater-for the possible risks and loopholes endangering security of smartphone users.

Using static passwords for user authentication is a risky venture. This is evident from the recent incidents of security infringement faced by major corporations. Around 6.5 million unsalted SHA1 hashed LinkedIn passwords were leaked in June 2012 [2]. A data breach in an FTP server owned by the IEEE resulted in leak of 0.1 million plaintext passwords in September 2012 [3]. Dropbox confirmed that it got hacked in July 2012 and therefore offered two-factor authentication from October 2012 [4]. Twitter, Skype, New York Times and Wall Street Journal suffered security breaches during the last one year [5]. Adobe said it was investigating how 150 million customer records were stolen during October 2013 [6]. Therefore, the recent trend is to shift towards TFA, which is more robust to security breaches and identity thefts. US Federal Financial Institutions Examination Council (FFIEC) recommends the banks to use TFA, in order to monitor monetary transactions [7]. The user credentials presented for remote validation for TFA schemes take a number of forms such as one time issued pass-codes, biometric traits, KeyFob hardware authenticators and digital certificates. In this work, we propose to use dynamic handwritten signatures in a TFA framework that runs on interactive hand-held devices.

Human biometrics can be defined as the automatic methods of recognizing different humans based on measurable anatomical, physiological and behavioral characteristics. Physiological biometrics are derived using invasive methods that are based on some physical parameters coming directly from human body. Non-invasive biometric traits that are characteristic of the concerned person are termed as Behavioral biometrics. We prefer to use behavioral biometrics (i.e., handwritten signatures) in the current work because of their high acceptability due to less cumbersomeness and ease in data collection. We argue that behavioral biometrics are more suitable to use in TFA systems because unlike the KeyFob tokens, user does not has to carry the issued identity all the times. Moreover, risk of identity loss/theft is negligible and these are difficult to replicate.

Among all the biometric measures, handwritten signature is an old, tested and most commonly used person authentication metric. Recent advances in sensing technologies and efficient touchable interfaces present in modern hand held devices have also made it an easily deploy-able authentication metric. Mobile devices are easily available to use and thus any authentication framework using biometric data collectible through these devices is of paramount importance. In contrast to traditional scanners and dedicated electronic devices for signature acquisition, newly available mobile devices are pervasive, equipped with high computational resources and also have extra features to attract consumers. Touch screen enabled smart phones and personal digital assistants (PDA's) can use signature verification for making on-line transactions, remote client authentication, signing legal documents and accessing various other online services. Signatures verification is possible both in static and dynamic modes. In current work, instead of using static signature data, we prefer to use dynamic signatures because of their better verification performance, robustness against forgery and ease in data collection through touch screen enabled hand-held devices [8].

Although mobile platforms pose a promising area for signature based biometric authentication, there are several

[1]This research work was carried out at Next Generation Intelligent Networks Research Center, Islamabad (http://www.nexginrc.org).

challenges associated as well. Restricted writing area, comparatively lower computational power of mobile devices and limitations in data collection[2] are the key challenges for achieving high performance [9]. The question of how the constraints put by hand-held devices on the user specific signature characteristics are studied in [10]. It has also been noted that device-to-device variability has a significant effect on the acquired signature dynamics and user specific traits as depicted in acquired data [11]. The high intra-user variability due to poor capturing conditions at different times can also not be ruled out [12]. The signing surface may not be familiar to many people who are accustomed to perform signatures on paper [13]. This in turn can have an impact on verification system's performance. It has also been reported for the case of hand-held devices that various dynamic features (such as, time, speed and acceleration) have relatively low discriminative power [11]. Time required to perform computations during training and testing is also high due to resource constraints.

The challenging acquisition scenario in the case of hand-held devices put constraints on achieving good verification performance. The performance of signature verification systems on hand held devices can not be evaluated on available datasets acquired using pen based tablets, hand glove or specialized signature pens, because of the difference in acquisition conditions. The users signing on capacitive touch screens of hand held devices normally use the tip of index finger. The finger contact can be lost during signing process, unclean touch screens can degrade sampled signal and natural factors like human skin sweating may also prove be a source of error. This acquisition scenario is challenging and completely different from subjects sitting on a chair and signing with pens or PDA stylus under controlled conditions. Biosecure signature evaluation campaign was launched in 2011 to test available online signature verification systems on two different evaluation tasks. Data was collected on a mobile platform (HP iPAQ PDA) as well and it was reported that the verification performance of systems decrease if data acquisition is carried out on a mobile device [14]. In view of these challenges, we propose a robust, lightweight signature verification system that can provide high performance and better user experience on hand-held devices.

The aim of this work is to propose a TFA system for use on mobile platforms like PDAs, smart phones, tablets or other touch screen based interaction devices. The TFA system uses inherence factor (*something characteristic to a user*) along with the knowledge factor (*something user knows*) during the authentication purposes. We use handwritten signatures in combination with passwords to provide a user friendly authentication framework. Such an implementation of proposed security framework is both reliable and flexible and it can adapted to various application scenarios in a *Software as a Service* (SaaS) paradigm. The major computational load of the aforementioned task is shifted on a cloud based application server so that a platform-independent user verification service with ubiquitous access becomes possible. Upto the best of our knowledge we are the first to propose and implement an online signature based verification system on smart phones that is linked seamlessly with the cloud. Custom applications are built for both the iOS and Android based devices which are linked with the cloud based two factor authentication (TFA) server. Using these smart phone based applications, we collect data from a group of users and test our scheme. Our system can also be used in conjunction with Security Assertion Markup Language (SAML) that enables multi-factor authentication by defining an open standard data format for exchanging sensitive data.

II. Cloud Based Two Factor Authentication Scheme

A. Overview

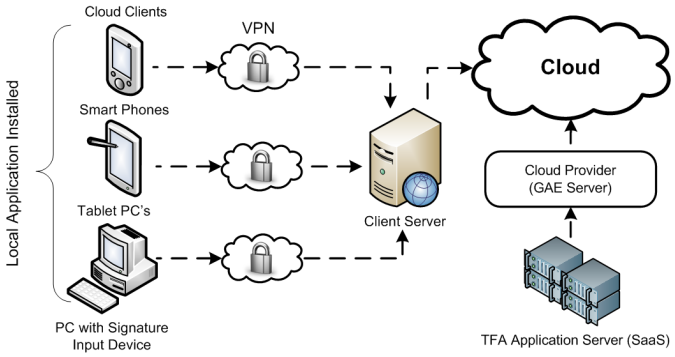
Our system is composed of a simple-to-use library that can be included in Android and iOS applications. A flexible architecture of our scheme provides user interface in the form of a locally installed smartphone application that connects with client server. Client server configuration is based solely on the application requirements and it may be hosting a SaaS, PaaS, IaaS, NaaS, STaaS, IDEaaS or an APIaaS[3]. This client server needs to be registered at our TFA application server. Upon registration, our Representational State Transfer (REST) web API issues a key to the client. This key is used along with the *Nonce* (number used once) and *cNonce* (client number used once) to authorize the client for using the services offered by our API. The user using the client services must also be registered at the client server. For this purpose the register/log-in screen at the user front end is shown in Fig. 1b. An overview of the scheme is shown in Fig. 1a.

During the registration/log-in process, the subject is required to enter his/her username, password, signatures and the client specific Uniform Resource Locator (URL). Acquired data is stored in the *.json* (JavaScript Object Notation) formatted secure data. This data format is language independent and provides human readable text based data interchange. The log-in action will redirect the user to the client server who has the authority to forward the user data to our TFA application server. The TFA application server will first validate the client using its issued key and *Nonce*, *cNonce* and then the decision regarding the authenticity of user will be made according to the supplied biometric data and password. The biometric data consists of dynamic handwritten signatures from the user. The system requires one time training so that a template of genuine biometric can be generated. This template is stored for authentication in future queries. The stored genuine template can be updated upon the request of client/user.

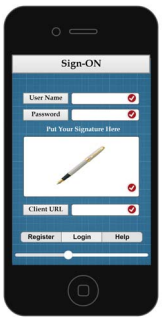
For the purpose of signature verification, two types of techniques are extensively used. These two major groups are Model based techniques and Distance based techniques. Model based techniques, such as Hidden markov models (HMM) [15] generate a stochastic model of user handwritten signatures. These stochastic models adapt with respect to user specific dynamics and represent a robust representation using probability distributions of features. Among the distance based approaches, dynamic time warping (DTW) [16] is the most popular one due to its flexibility and good performance over local features matching. We have used the DTW approach mainly because the model based techniques like HMM requires

[2]Only position information is available at lower sampling frequency compared to the dedicated signature collection devices

[3]software, platform, infrastructure, network, storage, an IDE or an API as a service respectively



(a) Overview of Complete System



(b) Register/Log-In Screen for Smart Phone User

Fig. 1: Multi-factor Authentication on Cloud

large test data, which does not suit our application situation [8]. The elastic template matching technique - DTW - is used to compare the probe biometric with the genuine template stored on the TFA application server. The client server is addressed regarding the outcome of authentication process. A detailed layout of our custom API is shown in Figures 2 and 3.

B. Implementation Details

Our RESTful API is based on a simple-to-use library for Android and iOS applications. The front end of application needs to be installed locally on each smart phone. The backbone of TFA system is implemented in cloud to access it as a web based service. Any client can make use of services of our RESTful API to authenticate its users. In this manner a seamless security and authentication framework is available to use by enterprises and businesses without getting in to the hassle of managing biometric data/passwords of its users.

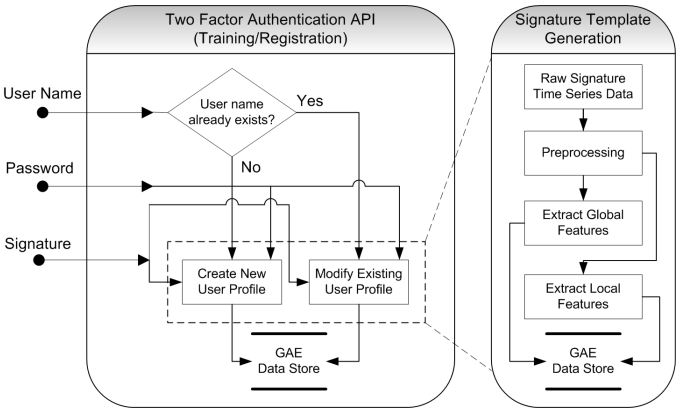


Fig. 2: Registration Process in TFA API

In our framework, we have used Google Application Engine (GAE) as the cloud services provider. GAE is a PaaS made available by the Google thanks to its huge data centers. This service supports cloud computing applications and equips application developers with the liberty to run web applications on Google's infrastructure. GAE supports common web technologies and allows persistent storage along with the queries, sorting and transactions. The applications are independent of physical resources of servers due to a sandbox environment which also ensures security and robustness. Automatic scaling and load balancing is also supported. As a result when the demand of a web service goes up, the allocated resources are also automatically upgraded [17].

GAE provides the opportunity to develop applications in three run-time environments: Java, Go, and Python. We preferred to use Python 2.7 development environment due to its simplicity and easy to use libraries e.g., NUMPY and MATH. An optimized sandboxed Python interpreter is available in GAE with the added support for some high level libraries and user friendly tools for managing web applications. We have used a lightweight 'webapp2' framework that is supported by GAE. This framework itself handles the details of the interface and let the programmer focus on application functionality. The flip side of the Python development environment is the several restrictions that are accompanied with it. First, the environment allows the execution of Python only code and thus any integrated snippet of code written in C is not supported. Secondly, any method which interacts or tries to modify external resources is prohibited. This behavior is a direct consequence of the sandbox environment in which the application runs.

We have used a thread-safe routine so that the different threads accessed by different users do not interact/modify each other. It can be called from multiple programming threads and potential race situations thus created are avoided with the collaborative access to shared data. For the purpose of storing user information, we have used GAE datastore that provides a schema-less object storage, with NoSQL flexibility. It also equips the developers with a query engine and atomic transactions. We have used High Replication Datastore (HRD) that stores data across various multiple data centers. Paxos algorithm [18] is used in this process and as a result we get promptly available reads and writes. The datastore is also equipped with Python datastore API that provides flexible storage platform for web based applications. This API offers unplanned downtime and multiple operations to be executed in a single transaction. Frequent and efficient read and write operations are available through this API and a consistent query answering is also provided [19].

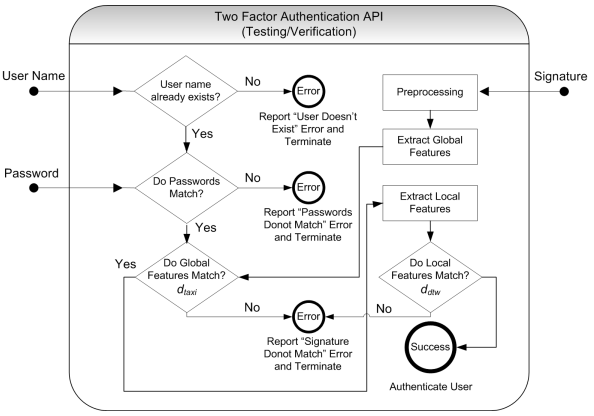


Fig. 3: Verification Process in TFA API

TABLE I: Global Features

Signature Duration (T)	Average Jerk $J_{avg} = mean(\dot{a_n}/\dot{t_n})$
Width (W) and Height (H)	Normalized Average Speed ($\frac{s_{avg}}{s_{max}}$)
Maximum Speed (s_{max})	Average X and Y Speed (s^x_{avg}, s^y_{avg})
Aspect Ratio ($A_r = W/H$)	Variance of Speed (s_{var})
Pressure Deviation	Maximum X and Y Speed (s^x_{max}, s^y_{max})
Average Acceleration a_{avg}	Average X and Y Acceleration a^x_{avg}, a^y_{avg}
Variance of Acceleration a_{var}	Variance of X and Y Acceleration a^x_{var}, a^y_{var}
Average Pressure p_{avg}	Maximum Pressure p_{max}
Maximum Acceleration a_{max}	Variance of X and Y Speed (s^x_{var}, s^y_{var})
Average Speed (s_{avg})	Point of Maximum Pressure
	Spread Ratio ($S_r = T/W$)

III. Signature Matching

In the proposed scheme, signature matching is performed by a hierarchical approach that depends both on global and local features. The first step in the template matching procedure is the preprocessing of raw times series data. Next, useful features are extracted and then a decision forest classifier is used to decide about the authenticity of probe biometric.

A. Preprocessing

The raw signature data is preprocessed to remove any angular or spatial shift. Center of gravity of each signature is set to zero and the average path tangent angle is also nullified by applying feasible amount of angular rotation. This alignment improves classification performance [20, 21].

B. Feature Extraction

During the feature extraction process, a number of global and local features are calculated using the time series signature data. In this manner a hybrid approach is used to extract useful discriminative information from signatures.

1) Global Features: The features extracted on the whole signature are termed as global features. A total of 27 global features are calculated. These are listed in Table I.

2) Local Features: Features extracted at each individual sampling point are called as local features and a pattern of feature values is generated. Local features used in our scheme are listed in Table II [20, 21]. Along with these local features, we include first and second order derivatives in the feature set.

TABLE II: Local Features

Feature Name	Formula
Sample Number n	$n = \{1, 2, 3..., N\}$
Time Stamp t	$t = \{t_{\text{start}} ... \ t_{\text{end}}\} = \{t_1 ... \ t_N\}$
X and Y Cartesian Co-ordinates	$\{x_n, y_n\}_{N\times 2}, \ \ \forall n \in [1, N]$
Speed along X and Y Axis	$\{s^x_n\} = \{\dot{x_n}\}/\{\dot{t_n}\}, \ \{s^y_n\} = \{\dot{y_n}\}/\{\dot{t_n}\}$
Root Mean Square Speed:	$\lvert s_n\rvert = \sqrt{{s^x_n}^2 + {s^y_n}^2}, \quad \forall n \in [1, N]$
Acceleration along X, Y Axis	$\{a^x_n\} = \{\dot{s^x_n}\}/\{\dot{t_n}\}, \ \{a^y_n\} = \{\dot{s^y_n}\}/\{\dot{t_n}\}$
Tangential Acceleration a_{tn}	$\lvert a_{tn}\rvert = \dot{s}_n = dif\left(\sqrt{{s^x_n}^2 + {s^y_n}^2}\right)$
Centripetal Acceleration a_{cn}	$\lvert a_{cn}\rvert = s_n.\dot{\theta}_n, \quad \forall n \in [1, N]$
Root Mean Square Acceleration	$\lvert a_n\rvert = \sqrt{{a_{tn}}^2 + {a_{cn}}^2} \quad \forall n \in [1, N]$
Path Tangent Angle	$\theta_n = tan^{-1}(\frac{\dot{y}_n}{\dot{x}_n}), \quad \forall n \in [1, N]$
Log of Radius of Curvature	$\delta_n = log(\frac{s_n}{\dot{\theta}_n}), \quad \forall n \in [1, N]$
X and Y Components of Curve	$\cos(\theta_n), \sin(\theta_n)$
Pressure	$p = \{p_1, p_2, p_3 ... p_N\}$

It must be noted that the derivatives involved in local features are calculated using a second order regression algorithm, which results in a better verification accuracy [15, 21].

C. Distance Measurement

The authentication decision is made by calculating two separate distance measures from local and global feature vectors of authentic and probe templates. These distance measures are then fed to a random forest classifier that predicts the class to which the probe biometric belongs i.e. a genuine or a forged signature. Based on this decision, a user is either allowed or denied access to the requested services by the client server.

1) Taxicab Distance for Global Features: This distance measure is used to quantify the change in the global features of genuine reference template and the query biometric template. It is defined as:

$$d_{taxi}(g^{ref}, g^{qry}) = \parallel g^{ref} - g^{qry} \parallel_1 = \sum_{i=1}^{N} |g_i^{ref} - g_i^{qry}|$$

where, g denotes the global feature vector. It must be noted that we preferred to use a translation invariant but rotation variant metric because of the nature of the template matching problem in our case [22].

2) Dynamic Time Warping for Local Features: DTW is an elastic matching technique that is used to make a comparison between two varying time series. DTW effectively minimizes the shifting in time and dynamically transforms the time axis. It uses dynamic programming to find the best path while aligning two time series. This best path maximizes the local match between two time series, however the cost of path gives the clue about mismatch between time series signals. The resulting dissimilarity index is used in the classification process to make a decision regarding the authenticity of signatures.

Suppose we have two arbitrary vectors which represents time series belonging to two different signature instances, $\mathbf{X} = [\mathbf{x}_1 ... \mathbf{x}_F] \in \mathbb{R}^{N_x \times F}$ and $\mathbf{Y} = [\mathbf{y}_1 ... \mathbf{y}_F] \in \mathbb{R}^{N_y \times F}$ where F is the total number of local dynamic features, N_x and N_y are the number of data points in $\mathbf{X}$ and $\mathbf{Y}$ respectively. A precondition of aligning data using DTW is that the $\mathbf{X}$ and $\mathbf{Y}$ must have equal sampling rates. Therefore, we resample both the signals if the sampling rates are different. At the beginning of the

matching algorithm, a distance matrix $\mathbf{U} \in \mathbb{R}^{N_x \times N_y}$ is built to store local pairwise distances between signatures $\mathbf{X}$ and $\mathbf{Y}$.

$$\mathbf{U} \in \mathbb{R}^{N_x \times N_y} : u_{i,j} = \| \mathbf{x}_i{}^T - \mathbf{y}_j{}^T \|_1$$

where $i \in [1 : N_x]$, $j \in [1 : N_y]$ and $\| \cdot \|_1$ is the ℓ_1 norm. DTW warps $\mathbf{X}$ and $\mathbf{Y}$ such that the cost or distance function is minimized over alignment path. The warping path $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, ...\mathbf{p}_m..., \mathbf{p}_M] \in \mathbb{R}^{M \times 2}$, (where $\mathbf{p}_m = [p_m^x, p_m^y]^T$, and $p_m^x \in [1 : N_x]$, $p_m^y \in [1 : N_y]$) must be followed while considering the boundary condition, monotonicity and continuity criterion. We have used a three step unconstrained policy function to define the continuity criterion:

$$\xi_u : \{(i+1, j), (i, j+1), (i+1, j+1)\}$$

The optimal path takes M steps to match both the series with least cost. This path cost can be denoted as:

$$\eta = argmin \left(\sum_{m=1}^{M} \| \mathbf{x}_{p_m^x}^T - \mathbf{y}_{p_m^y}^T \|^2 \right)$$

$\mathbf{X}$ and $\mathbf{Y}$ can be aligned in a number of ways, exponential in N_x and N_y. Dynamic programming provides an efficient approach ($O(N_x N_y)$ time) to reach the desired minimum cost path.

D. Classification

Following the computation of distance measures for both the local and global features, a random decision forest classifier (with 50 trees) is employed. The metric calculated on global features is efficient in time but gives a crude estimate about the genuineness of signature. Therefore it is put earlier in the decision hierarchy with a loose bound. Distance metric calculated on local features is put in the lower hierarchical level so that a more detailed analysis can be made regarding the authenticity. The signature that match well on both the comparison levels is declared genuine. The strictness of match can be set by the user through the interface. A more strict policy regarding signature match will need a high level of similarity between the reference and probe biometric.

IV. Experiments and Analysis

A. Dataset

As described in Sec. I, the research problem of online signature verification on smart phones has to deal with many challenging scenarios. Therefore, it will not be fair to evaluate a smart phone based verification system can not be made on a dataset collected using digital tablets or specialized devices which have better data collection conditions. In view of this need, we have collected a dataset of our own using only smart phones. A variety of phones were used during the process of data collection so that a diverse dataset - independent of device specifications - can be made possible. We used three smart phones (from Apple, Samsung and Sony) whose specs and listed in Table III. All three devices come with capacitive touch screens and subjects used their index finger to perform signatures. Custom made applications were installed on phones and a total of 20 users were enrolled on the cloud based service[4].

[4]The data gathered during this process will be make publicly available on the author's website soon after acceptance of this manuscript.

TABLE III: Device Specifications

Specifications	Galaxy GioS5660	Xperia U	iPhone 4
Brand	Samsung	Sony	Apple
CPU Speed	800 MHz	1 GHz (dual core)	1 GHz
CPU Type	ARM-11	ARM Cortex-A9	ARM Cortex-A8
OS	Android v2.2 (Froyo)	Android v2.3 (Gingerbread)	iOS 6.1
GPU	Adreno 200 GPU	Mali-400	PowerVR SGX535
Touch Screen	—- TFT capacitive, Multitouch —-		
Screen Size	320 x 480 pixels, 3.2 inches diagonal	480 x 854 pixels, —— 3.5 inches diagonal ——	640 x 960 pixels,
Sampling Rate	——— 60 Hz Maximum ———		

TABLE IV: Resource Utilization

Resources	Consumption
Number of Instance hours in test	1 instance hour
Number of registration requests	20 users
Number of verification requests	1400 requests
Average outgoing bandwidth per request	193 bytes per request
Average incoming bandwidth per request	7.5 kilobytes per request
Number of datastore write operations	0.04 million
Number of datastore read operations	0.0015 million
Average application response computation latency	124 ms
Average application instance memory usage	28.69 MB
Average datastore entry size (single user's data)	22 KB
Average cost of adding a new user	1.855×10^{-3} USD one time $+ 4 \times 10^{-6}$ per month
Average cost of a user's MFA operation	5.6×10^{-5} USD one time

B. Results

To test the application's resource consumption and measure scalablilty in terms of cost, an experiment was performed where the cloud application was utilized by a set of 20 users through an automated script. The test was run for one full instance hour of the application. In total, 20 users were registered to the system, and 1400 multifactor authentication operations were performed during this experiment. The resource utilization during this test was computed and recorded by GAE servers. The results are given in Table IV. Based on the GAE pricing [5], we performed a cost-analysis of the system (per user) to illustrate the scalability of the system when new users are added. The average cost of adding a new user is very low (0.001855 one time + 0.000004 per month). The average cost of performing the multi-factor authentication process on a registered user is also very low (0.000056 one time). This makes this system very scalable for use in applications having a large user base.

Five signatures were collected from each user during the

[5]retrieved on Dec 12, 2013 from the url: https://developers.google.com/appengine/pricing

TABLE V: Illustrating Datastore Entry Decomposition (22 KB per entry)

Name	Type	Size	Percentage
Password	String	17 Bytes	0.08%
SignatureGlobal	Float	660 Bytes	3.00%
SignatureLocal	Float	19 KBytes	86.36%
Username	String	17 Bytes	0.08%
Metadata	-	2 KBytes	9.09%

TABLE VI: Signature Verification Accuracy

FAR	FRR	ROC Area	F-Measure	Accuracy
0.92%	16.9%	0.981	98.4%	98.4%

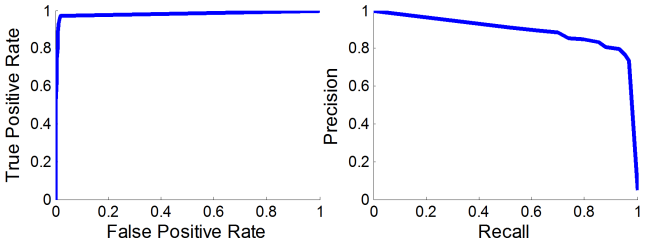


Fig. 4: ROC and Precision-Recall Curves for the Signature Verification Task

enrollment phase. The random forest classifier was trained on the genuine samples from each user. The samples supplied from the other users are treated as a random forgery attempts during the training process. Table VI shows the overall verification performance of our framework. The verification scheme is robust to forgery attempts with a low FAR (0.92%). The FRR is relatively higher (16.9%) mainly because the genuine signatures have to face the restrictive data collection conditions (see Sec. I). It must be noted that the accuracy, 98.4%, shows the security strength when only signatures are used. In our implementation, user specific passwords are used in combination of signatures and thus the resulting security strength is better. During the experiments, local features proved to be more accurate than the global features during the authentication process. However, they were more expensive (both in terms of time and memory) to calculate as compared to global features. Therefore, the authentication decision is made in a hierarchical fashion where local features are calculated once global features match reasonable well with the genuine template. Nonetheless, it does not take more than a few seconds to validate a genuine signature attempt. Fig. 4 shows the ROC and Precision-Recall curves.

V. Conclusion and Future Work

We propose a cloud based two factor authentication scheme that combines human biometrics and knowledge factor to enhance security. The proposed scheme is easily scalable and is available to use on mobile platforms such as smart phones and PDA's. The cost and resource requirements of the proposed SaaS are low and independent of the user end platform. We show that the implemented system performs well for a relatively small group of users. In future, we will evaluate our authentication framework on a larger-scale, with more clients and users registered on the cloud based service.

References

[1] A. Allan, "Magic quadrant for user authentication," 2012.

[2] P. Kamp, P. Godefroid, M. Levin, D. Molnar, P. McKenzie, R. Stapleton-Gray, B. Woodcock, and G. Neville-Neil, "Linkedin password leak: Salt their hide," *Queue*, vol. 10, no. 6, p. 20, 2012.

[3] R. Dragusin, "Data breach at ieee.org:100k plaintext passwords," http://ieeelog.com, 2013, [Online; accessed 22-Oct-2013].

[4] J. Brodkin, "Dropbox confirms it got hacked, will offer two-factor authentication," http://arstechnica.com/security/2012/07/, 2012, [Online; accessed 22-June-2013].

[5] B. Lord, "Keeping our users secure," https://blog.twitter.com/2013/keeping-our-users-secure, 2013, [Online; accessed 28-Nov-2013].

[6] P. Ducklin, "Anatomy of a password disaster - adobe's giant-sized cryptographic blunder," http://nakedsecurity.sophos.com/2013/11/04/, 2013, [Online; accessed 22-Nov-2013].

[7] FFIEC. (2005, Feb.) Ffiec releases guidance on authentication in internet banking environment. [Online]. Available: http://www.ffiec.gov/press/pr101205.htm

[8] S. H. Khan, M. A. Akbar, F. Shahzad, M. Farooq, and Z. Khan, "Secure biometric template generation for multi-factor authentication," *Pattern Recognition*, vol. 48, no. 2, pp. 458–472, 2015.

[9] D. Impedovo, G. Pirlo, and R. Plamondon, "Handwritten signature verification: New advancements and open issues."

[10] D. Simpsons, R. Spencer, and S. Auer, "the effects of constraining signatures," *Journal of the American Society of Questioned Document Examiners*, vol. 14, no. 1, pp. 39–50, 2011.

[11] S. Elliott, "Differentiation of signature traits vis-a-vis mobile- and table-based digitizers," *ETRI journal*, vol. 26, no. 6, pp. 641–646, 2004.

[12] M. Martinez-Diaz, J. Fierrez, J. Galbally, and J. Ortega-Garcia, "Towards mobile authentication using dynamic signature verification: useful features and performance evaluation," in *ICPR*. IEEE, 2008, pp. 1–5.

[13] M. Martinez-Diaz, J. Fierrez, and J. Ortega-Garcia, "Incorporating signature verification on handheld devices with user-dependent hidden markov models," in *Int. Conference on Frontiers in Handwriting Recognition*, vol. 32, 2008.

[14] N. Houmani, S. Garcia-Salicetti, B. Dorizzi *et al.*, "Biosecure signature evaluation campaign (esra'2011): evaluating systems on quality-based categories of skilled forgeries," in *International Joint Conference on Biometrics*. IEEE, 2011, pp. 1–10.

[15] J. Fierrez, J. Ortega-Garcia, D. Ramos, and J. Gonzalez-Rodriguez, "Hmm-based on-line signature verification: Feature extraction and signature modeling," *Pattern Recognition Letters*, vol. 28, no. 16, pp. 2325–2334, 2007.

[16] R. Martens and L. Claesen, "On-line signature verification by dynamic time-warping," in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, vol. 3. IEEE, 1996, pp. 38–42.

[17] Google, "Google application engine for developers," https://developers.google.com/appengine/, 2013, [Online; accessed 25-Jan-2013].

[18] T. Chandra, R. Griesemer, and J. Redstone, "Paxos made live-an engineering perspective (2006 invited talk)," in *Proceedings of the 26th ACM Symposium on Principles of Distributed Computing-PODC*, vol. 7, 2007.

[19] Google, "Gae datastore."

[20] J. Fierrez-Aguilar, S. Krawczyk, J. Ortega-Garcia, and A. Jain, "Fusion of local and regional approaches for on-line signature verification," *Advances in Biometric Person Authentication*, pp. 188–196, 2005.

[21] S. H. Khan, Z. Khan, and F. Shafait, "Can signature biometrics address both identification and verification problems?" in *ICDAR*. IEEE, 2013, pp. 981–985.

[22] J. Yu, J. Amores, N. Sebe, P. Radeva, and Q. Tian, "Distance learning for similarity estimation," *TPAMI*, vol. 30, no. 3, pp. 451–462, 2008.