2015 IEEE 8th International Conference on Cloud Computing

User-Friendly and Secure Architecture (UFSA) for Authentication of Cloud Services

Reza Fathi*, Mohsen Amini Salehi†, and Ernst L. Leiss*
*Computer Science Department, University of Houston
{rfathi, coscel}@cs.uh.edu
†Computer Science Department, University of Louisiana at Lafayette
amini@louisiana.edu

***Abstract*— Clouds are becoming prevalent service providers because of their low upfront costs, rapid application deployment, and high scalability. Many users outsource their sensitive data and services to cloud providers. Users frequently access these sensitive services through devices and connections that are vulnerable to thieving and eavesdropping. Therefore, users are desperate of robust security measures to protect their data and services privacy in clouds. In particular, robust authentication techniques are demanded by users for safe access to cloud services. One technique is to utilize multiple authentication factors (a.k.a multi-factor authentication) to access cloud services. However, the challenge is that the multi-factor authentication technique is not effective as it causes user frustration and fatigue. To address this challenge, in this study, we propose a multi-factor authentication architecture that aims at minimizing the perceived authentication hardship for cloud users while improving the security of the authentication. To achieve the goal, our authentication architecture suggests a progressive manner to leverage access to different levels of cloud services. At each level, the architecture asks for authentication factors by considering the perceived hardship for users. To increase the security and user convenience, the architecture also considers implicit authentication factors in addition to the explicit factors. Our evaluation results indicate that authentication using the proposed architecture decreases the users' perceived hardship up to 29% in compare with other methods. The results also reveal that our proposed architecture adapts the authentication difficulty based on the user condition.**

***Index Terms*—Cloud services, User-friendly authentication, Multi-factor Authentication, Sand-boxing**

I. Introduction

Third party cloud service providers let their users to obtain on-demand services. This paradigm is flourishing in the computing industry because it reduces the upfront costs and deployment time of applications. However, sharing resources with other users on the cloud premises, vulnerable devices and network connections have arisen security concerns among the cloud users [13, 24, 31, 36]. To address these concerns, cloud providers are seeking for stronger security solutions at different levels to assure their users about the security of their offered services.

More specifically, with the increasing connectivity of mobile devices to the Internet, many sensitive operations (e.g. financial transactions) are carried out using these devices. In addition, heaps of sensitive data, hosted on the cloud servers, are accessed by hand-held and mobile devices such as smartphones. These devices, in particular, are vulnerable to security threats such as being stolen. Moreover, lack of users' trust in the cloud security, and leakage of confidential data or services are other security challenges of cloud providers [22]. Therefore, an ideal security model for cloud should provide an acceptable trust to users and guarantee that there will be no unauthorized access to their data and services.

Authentication is the process of determining whether or not a user is the one claiming to be. It also provides a security level that determines the confidence degree of a system on the authenticated user. Authentication is a crucial step in the security suit that directly involves the users of a system. To have a robust authentication, it is a common practice to apply multiple authentication factors (a.k.a. *multi-factor authentication*) to grant access to sensitive services [5]. As a result, the security improves because it does not depend on a single factor. For example, if a bank card is stolen, the PIN number is still required to cash the money.

Using multiple authentication factors is burdensome for users. Recent research works have revealed that convenience is a major factor for users in preferring one method over another [3]. In fact, users prefer to utilize convenient authentication methods rather than sophisticated ones that are typically more secure [18]. This issue usually causes sophisticated authentication methods to fail in practice, in spite of their higher security. An alternative solution proven to be less difficult for users is *progressive multi-factor authentication* [29]. It applies the authentication factors in a progressive manner upon users' demand to access more sensitive services.

Another authentication approach that relieves the user from authentication is *implicit authentication* where patterns of the user behavior is used for authentication. This is opposed to *explicit authentication* where the user has to directly provide the authentication factor. Nonetheless, although implicit authentication methods provide user convenience, they cannot provide a sufficient security level due to their high false-positive rate [6, 10, 12, 33].

The challenge is that convenient authentication methods are not secure and sophisticated authentication methods are not effective as they are not user-friendly. In fact, any solution for this challenge should satisfy both the security and usability aspects at the same time. Progressive authentication can be helpful, however, when it is applied frequently, it also results in user fatigue. Therefore, the research question that we investigate in this study is: *how can we minimize user per-*

2159-6190 2015

DOI 10.1109/CLOUD.2015.75

ceived hardship with progressive multi-factor authentication in accessing cloud services?

To answer this question, we propose an architecture based on the implicit and explicit authentication factors. We also define a measure for hardship that enables us to estimate the difficulty of a given set of authentication factors. Then, we utilize the defined measure to propose an algorithm that minimizes the perceived authentication difficulty for the user at each access level. Our proposed algorithm also operates adaptively based on the user condition. That is, if the user cannot pass the implicit authentication, then the algorithm introduces more difficult authentication factor to her. On the contrary, the more implicit authentication can be passed, the less difficult authentication factors is required from user.

Our proposed architecture (UFSA) introduces a novel method for selecting explicit authentication factors in a multi-factor authentication method to make it more user-friendly and usable without sacrificing the security level. In summary, this study includes the following contributions:

- Proposing an architecture based on progressive authentication to access cloud services.
- Selecting a subset of explicit authentication factors at each authentication level that minimizes the perceived user authentication hardship.
- Analyzing the performance of UFSA, in terms of adaptability and user perceived hardship, in various working conditions.

Our evaluation results show that UFSA reduces users perceived authentication hardship remarkably in compare with methods that do not consider user hardship in the authentication process. In addition, the evaluation results reveal that UFSA adapts the authentication process based on the user condition. That is, it increases the authentication hardship (i.e., asks for more authentication factors) in case implicit authentication factors of the user are not valid. Accordingly, it eases the explicit authentication when more implicit factors are valid.

The rest of this paper is organized as follows: In Section II related research work is introduced. Next, in Section III, sand-boxing method is explained. Then, in Section IV, the proposed authentication architecture is described. Performance evaluation of the proposed architecture is reported in Section V. Finally, conclusion and future works are provided in Section VI.

II. Related Work

There are two types of user authentication namely, explicit and implicit. Explicit authentication requires user involvement to identify and verify the subject requesting a service. In [4], the authors discuss the logic of authentication and consider four important authentication protocols and describe their weak points. The author of [14] describes the security risks of authentication methods depending on synchronized clocks.

With the increase of security concerns, researchers have suggested to use more than one authentication method to achieve a higher security. Duncan discusses the main factors of a dual-factor authentication system which should be considered during the decision-making process [2]. He highlights some of the most important issues to be considered before selecting an authentication solution. In [16], a dual-factor authentication method using the voice of the user has been introduced. This method combines single-factor voice verification with token technology. It uses pre-recorded speech and a text-to-speech voice cloning technology to improve the strength of the authentication process. In addition, its generated spoken one-time passcode is not vulnerable to voice cloning or reply attacks.

In [5], a multi-layer and multi-factor authentication method is provided for intranet, extranet, and internet users to access a web main system. In this system, intranet users only pass through the conventional user name and password authentication process to access the authenticated network resources. The extranet users, however, are required to pass one more level of authentication to prove their identity. The Internet users face the highest authentication complexity to be segregated form the invalid users. Implicit authentication has the advantage of not involving the user in the authentication process. It is done in the background and tries to identify and authenticate users using biometric or behavioral parameters. This method has obtained many applications recently, due to the increasing usage of hand held and mobile devices.

In [8], a biometric approach by ear shape recognition is proposed. The idea is using smart-phones to learn a pattern from the geometric features of the ear of the users. These features are used to learn patterns for implicit authentication of a user.

In [7], a method for implicit authentication by using the pattern of inputing user name and password is proposed. This is in addition to a layer for the explicit authentication to increase the security. Shi et al. [33] propose implicit authentication through learning user behaviour based on his messaging, phone calls, browsing history, and device location. Sandnes and Zhang studied feature extraction for learning from device holding hand, stroke size, timing, speed, and timing regularity [32].

Continual implicit authentication on mobile devices is proposed in [1, 7, 8, 10, 19, 20, 23, 29]. They analyze typing motion behavior of the users to learn. The method in [23] considers entering text on the keyboard while the users' finger direction movement is considered. In [29], a progressive authentication method on mobile phones is proposed. It tries to facilitate the authentication process by decreasing authentication times. It uses both implicit and explicit factors to attain the authentication score which is required in each step. In [9], continual mobile authentication using touchscreen gestures gathered by a digital sensor glove is proposed. This glove collects more gesture information, which is used to achieve implicit authentication.

These implicit authentication methods are used as a direct authentication method, mostly on the touchscreen mobile devices. Since they have relatively high false acceptance rate, they are not dependable for the environments such as clouds which host sensitive data and services. In our approach, these implicit methods are used as a factor to give weight to the score of explicit authentication methods. That is, in our

proposed architecture, these implicit methods are used as an auxiliary factor.

III. Sand-boxing

Since cloud services are provided based on shared resources, any code in the clouds should be run in isolation and with attention to security. Sand-boxing can provide isolation by preventing dangerous actions of malicious codes or destructive data thieving [21, 34]. That is, it provides a security layer by implementing isolation between users and processes [25]. Sand-boxing is widely applied and is able to provide an acceptable security [11]. The level of isolation provided by sand-boxing depends on its static or dynamic configuration.

Static sand-boxing is to configure the sandbox at its initialization time. Although static sand-boxing needs a simple configuration, it is restrictive and is not suitable for dynamic and shared environments. To apply static sand-boxing, the administrator should decide what subset of the system should be protected beforehand. This decision will affect the whole system without any flexibility to change it. The inflexibility makes static sand-boxing inappropriate for cloud environments.

Configurable sand-boxing is a solution which offers fine-grained control to users to define which actions are allowed and which are not. An instance of configurable sand-boxing is in Java 2 [35], in which users can configure the permissions and trust they decide to offer to a running code. A code configured as more trusted, can perform more privileged operations. Not only does any misconfiguration cause many security issues, just as in static sand-boxing, but it also needs high technical skills to configure it [11]. In other words, it is hard and seems very complicated to the typical users who know little about security configurations.

Although defining a proper sand-boxing is difficult, dynamic sand-boxing can automatically learn security policies [17, 27]. The learning happens at the training phase in which the learning security policy module runs and records all the permissions needed to access a service or a file.

In this study, operation sensitivity of a service is determined by the permissions required to access that service. In any access, a user should provide an authentication score equal to or higher than the required operation sensitivity to be able to access the service. This dynamism suites security controls of cloud systems that need more flexibility at defining and applying the policies.

IV. Progressive User-Friendly Authentication

Authentication is the process of determining whether a connecting subject is the one claiming to be. Authentication can depend on the factors a user knows, has, is, or which belong to the user [26, 28]. There are two kinds of authentication, explicit and implicit. Explicit authentication requires user direct involvement in the process of authentication, such as entering user-name password, PIN code, or using a swipe card. This type of authentication imposes a burden on the users, hence, causes user frustration when it is demanded frequently. Implicit authentication tries to authenticate a user without her direct involvement, therefore, it is not burdensome and is welcomed by the user. Some implicit authentication methods operate based on the user characteristics (e.g., biometric or behavioral patterns such as touching or typing patterns) or the device characteristics (e.g., the GPS location or the IP address of the connecting device) to authenticate users. Depending only on the implicit authentication methods is a security risk, as these methods have relatively high false positive rate [7, 10, 33]. To avoid the user fatigue resulted by explicit authentication, implicit authentication can be used in combination with the explicit methods [12, 15].

In this paper, we propose an architecture, shown in Figure 1, that utilizes implicit authentication along with the explicit ones. The architecture includes five main components. Sand-boxing component performs user access control to different service levels in the Cloud. Explicit and implicit authentications are the set of authentication factors that the user can exercise to gain access to different levels of Cloud services. Meta-learner is a machine learning engine that provides an authentication weight based on the implicit authentication factors. Component F calculates the current authentication score of a user and determines the optimal set of explicit authentication factors (i.e., with minimum user perceived hardship) that a user should exercise to gain access to a higher service level.

In the architecture, each explicit authentication factor is assigned an *authentication score*. That is, a user obtains authentication score as she presents (i.e., validates) more authentication factors and zero for unpresented ones. Accordingly, each cloud service is assigned an *operation sensitivity* that indicates the security criticality of that service. The operation sensitivity is configured by the system administrator or by using a dynamic sand-boxing (see Section III). When a service is highly sensitive, it is assigned a higher operation sensitivity, meaning that a higher trusted user can access the service. A user gains a higher trust (and access level) by obtaining a higher authentication score.

We utilize implicit authentication factors to give weight to the score obtained by explicit authentication factors. As such, when implicit authentication factors are validated (invalidated), the score given to explicit methods will get higher (lower) weights. This is achieved by function F in Figure 1. For instance, using a known IP address for authentication can validate the implicit authentication, as opposed to using an unknown IP address (e.g., through connecting form a public place) invalidates the implicit factor.

Let p_{i} the authentication score of explicit factor i that a user can utilize for authentication. Also, let w the implicit authentication factor which returns a value in $-1 \leq w \leq 1$. The negative extreme (i.e., when $w = -1$) indicates that the implicit authentication has no trust on the user whereas the other extreme (i.e., when $w = 1$) indicates that the implicit authentication has a full trust on the user. The value of w is obtained from the implicit factors by the meta-learning engine. In fact, it exploits features of impilicit factors to build a two layered machine learning engine. First layer includes machine learning methods such as Support Vector Machines; and the second layer is a meta-learner which provides output of

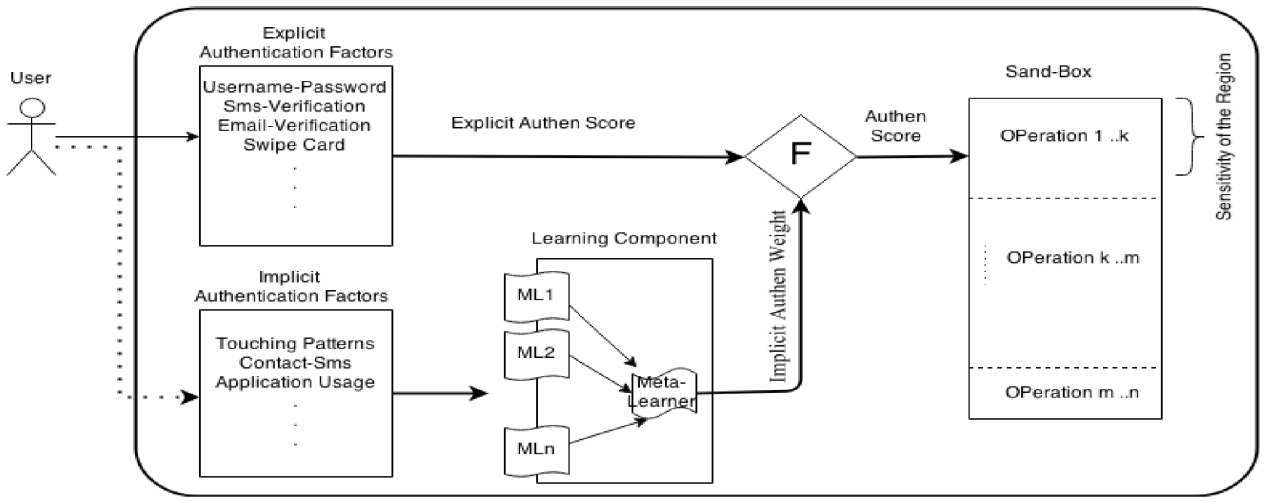


Fig. 1: Progressive and user-friendly authentication architecture: Implicit authentication factors weight the score of explicit authentication methods which should be higher than or equal to the operation sensitivity of the sandbox region.

the learning component. For this research, we prototyped the meta-learning component as a black-box to return simulation values for w and left investigating it as a future work. Getting w from the learning engine, function $f(w) = w + 1$ is used to calculate a weight cofactor as input of component F.

Equation 1 formulates how the authentication score for a set of explicit authentication factors, denoted L_s, is calculated by F. As we can see in Equation 1, based on the value of implicit authentication, $f(w)$ gives zero to double weight to the explicit authentication scores.

$$L_s = f(w) \cdot \sum_{\forall i \in L} p_i \tag{1}$$

In the event that a user wants to access a cloud service v, her obtained authentication score is compared against the operation sensitivity of that service (ϕ_v). A user is able to access service v if and only if $L_s >= \phi_v$. Otherwise, the user has to increase her authentication score by at least $g = \phi_v - L_s$.

Finding a subset of available authentication factors to fill the required gap (g) and, at the same time, minimizing the hardship of the authentication is challenging. We developed Algorithm 1 to find the subset of available authentication factors to fill the authentication score gap (g) and enable the user to access a given cloud service with minimal perceived hardship. We consider the authentication score of each factor and its hardship in the algorithm. The required gap (g) and the tree (T) that explores all cases are two inputs to Algorithm 1. The other input is the set of available authentication factors (shown as L). For a given authentication factor $i \in L$, we calculate the *fitness ratio* as p_i/h_i, where h_i is the perceived hardship for authentication factor i. The elements of L are sorted based on the fitness ratio in the descending order.

As mentioned earlier, to gain access to service v, the total authentication factor score obtained by a user, should be equal or higher than the operation sensitivity of service v. The algorithm considers all possible combinations of the authentication factors. However, for the sake of time efficiency,

Algorithm 1: Selecting a subset of authentication factors with the minimum user perceived hardship to obtain access to a cloud service.

Input: int g, Tree T, List L
Output: Path $best$

```
1  Queue Q;   //Create an empty Queue
2  Node u, r, best;
3  r ← best ← root(T);
4  enqueue(Q, r);
5  while not empty(Q) do
6      dequeue(Q, r);
7      foreach u childof r do
8          if boundScore(u) ≥ g then
9              Enqueue(Q, u);
10             bu ← boundHardship(u);
11             bb ← boundHardship(best);
12             cu ← boundCount(u);
13             cb ← boundCount(best);
14             if (bu < bb) OR (bu = bb AND cu < cb) then
15                 best ← u;
16 return best;
```

it only explores promising branches that potentially can fulfill the score gap and provide the required authentication score. If a promising branch has less perceived hardship than the current best solution, it is replaced as the best found solution. As a result, the proposed algorithm returns the set of authentication factors with the minimum perceived hardship. We formally define the perceived hardship for a set of authentication factors (L_h) based on Equation 2. That is, the perceived hardship for authentication factors in L is the sum of hardships of the authentication factors (h_i) in L.

$$L_h = \sum_{\forall i \in L} h_i \tag{2}$$

We use a tree data structure to encompass all possible cases in Algorithm 1. At the root of the tree we assume no authentication factor is selected. Then, at each level of the tree, we decide about inclusion or exclusion of one authentication factor from L. We use the $Queue$ data structure to hold unexplored nodes of T, shown as Q in Algorithm 1. We start by enqueuing the root r (step 4). Then, while there is an elements in Q, we dequeue r (step 6) and enqueue its children u, if they are promising (steps 8, 9). To check if node u is promising, before enqueuing it, we evaluate its potential to be a feasible response (step 8). In fact, function $boundScore(u)$ determines the total authentication score obtained by the path from root to node u and by including all remaining authentication factors from node U in T.

Next, in step 15, we update the $best$ found solution, if node u has less hardship than the current best solution. Function $boundHardship(u)$ uses Equation 2 to calculate the the overall perceived authentication hardship from the root to node u and promising nodes from u. It is worth noting that, if node u results into the same authentication hardship as the current best solution, it is selected as the $best$ solution, if and only if, it includes fewer authentication factors. Function $boundCount(u)$, in steps 12 and 13, determines the number of authentication factors involved in the path from root to node u and promising nodes from u. This tries to minimize the number of authentication factors required to fill gap g.

It is noteworthy that $best$ indicates a path from root to a leave in T. Following the path determines the subset of authentication factors in L that has the minimum hardship.

V. Evaluation

Our goal is to minimize the users' perceived hardship during the use of Cloud services. Hardship is the users' feeling of inconvenience when trying to pass an authentication step. We rank hardship values [30] of each authentication factor between 1 and 100 as the inverse of the easiness a user encounters. When a method seems harder, a higher value will be assigned for its hardship. Administrators also determine an authentication score for each explicit authentication factor with values between 1 through 100; while hardship of the methods will be assigned by the users based on their preference and feeling of the hardship. In other words, each user determines how hard doing an authentication step is for her, which are subjective numbers. For example, a user may set 10 and 30 for pin-code and swipe card respectively while another user may assign 40 and 20 for these factors based on her preferences.

For evaluation we compare the hardship scores of different authentication methods. We implemented two other solutions that serve as baseline solutions. The Naïve method selects the easiest available method at each step regardless of the methods authentication score. The Greedy method selects an available authentication factor with the highest fitness ratio of p_i/h_i at each step. We implemented an emulation to do a proof of concept. To demonstrate the performance of UFSA from different angles, we first provide two case studies and then we show the simulation results.

A. Case Study 1:

Region Name	Operations List	Region's Sensitivity
Reg1	Operation1	10
Reg2	Operation2	20
Reg3	Operation3	40
Reg4	Operation4, Operation5	60

TABLE I: Defined sections of a sandbox

Operation	Sensitivity Score
Operation1	10
Operation2	20
Operation3	40
Operation4	60
Operation5	80

TABLE II: Defined operations' sensitivity

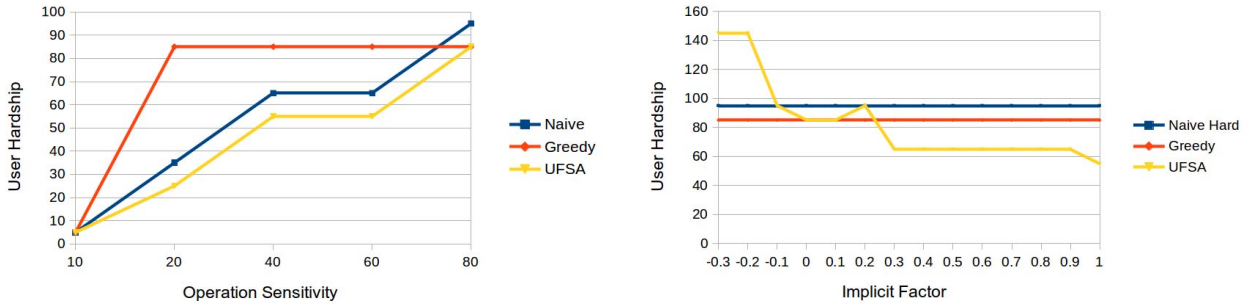
We defined a sandbox having four regions with the defined regions' border sensitivity and the operations belonging to them in Table I. In Table II, we defined the explicit operation sensitivity of the five operations defined in Table I. In Table III, a list of explicit authentication methods supported by the cloud system for a user is defined.

Figure 2a shows hardship faced by the users while increasing the operations' sensitivity. In this figure, it is shown that the Greedy method causes higher hardship by acting greedy and selecting a method with the highest fitness ratio at each step. Our proposed method (UFSA) in Algorithm 1 causes less hardship by selecting the easiest methods which provides at least the required authentication score. The Naïve method causes higher hardship comparing to UFSA by selecting only the easiest method at each step.

Figure 2b shows the accumulated hardship obtained by applying different methods. Since the Naïve method selects the easiest factor at each step, it often results in to the highest hardship. UFSA leads to more hardship than the Naïve method when the implicit weight w is negative. In fact, implicit factors weight the score that explicit methods provide and scales the scores down when bad events happen. In contrast, when the implicit factors are validated, implying that good events are happening, the user gets a higher weight of w (see Equation 1) for her authentication factors. So, each authentication step gains a higher score causing the user faces less hardship during the authentication process. The figure shows that UFSA causes less hardship comparing to both other methods when good events occur. The Greedy method causes higher hardship than UFSA. It chooses a method with a higher fitness ratio, however with a higher hardship.

B. Case study 2:

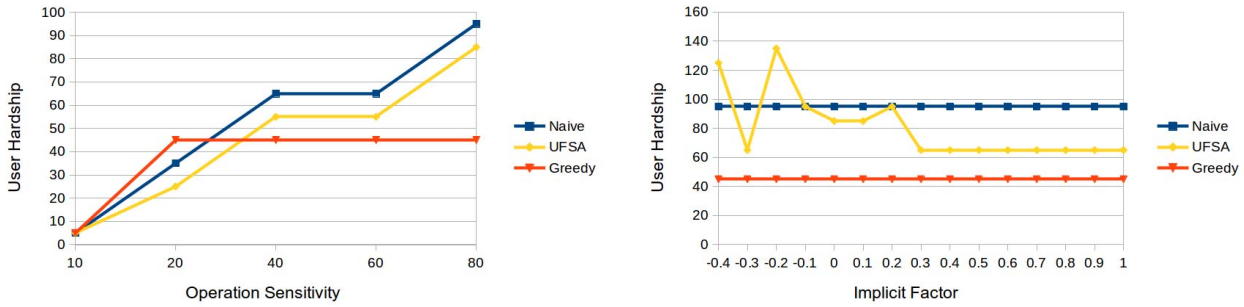
We provided another example in Table IV including operations' sensitivity to compare the behaviour of UFSA in a biased example. It is called biased because the user set 40 as the hardship of a method which has the highest score. In other words, its fitness ratio is 2.25 while other factors have the ratio of less than or equal 1. Figure 3a shows that, in this case, the Greedy method can cause less hardship comparing to the UFSA. Since the UFSA selects the authentication factors in a progressive way deciding at each authentication step, it chooses a method that is the best choice at the case level



(a) Impact of operation sensitivity on the user hardship

(b) Impact of implicit factors on the user hardship

Fig. 2: Evaluating perceived user hardship referring to Tables II and III.



(a) Impact of operation sensitivity on the user hardship

(b) Impact of implicit factors on the user hardship

Fig. 3: Evaluating perceived user hardship on a biased setting referring to Tables II and IV.

Number	Name	Score	Hardship
1	Bio Question	5	10
2	Password	10	5
3	Sms	20	20
4	Voice	30	30
5	Image	30	30
6	Token	40	50
7	Attend	90	80

TABLE III: Supported explicit authentication of a user

Number	Name	Score	Hardship
1	Bio Question	5	10
2	Password	10	5
3	Sms	20	20
4	Voice	30	30
5	Image	30	30
6	Token	40	50
7	Attend	90	40

TABLE IV: A biased configuration for the supported explicit authentication of a user

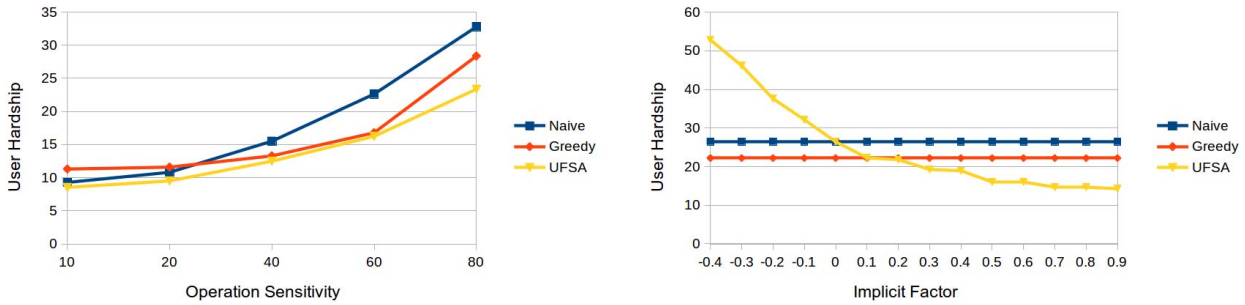
and moment. However, it doesn't guarantee to select the best global subset if the user later proceeds to more sensitive operations requiring a higher authentication score. Indeed, it is unknown beforhand that how many sensitive operations a user will request to perform during a session. The Greedy method chooses the best greedy option at each step which is the optimal greedy choice in general too.

C. Evaluating User Perceived Hardship:

Figure 3b shows that UFSA causes higher accumulated hardship when the implicit factor is negative because of the bad events which cause suspicious weighting of the authentication factors. When good events happen, our method causes less hardship comparing to the Naïve method. Nevertheless, the Greedy method causes less accumulated hardship in a biased example when the user continues to do all the sensitive operations defined in the operations table.

However, in simulation test running, UFSA outperformed the Naïve and Greedy methods by achieving less hardship. In each running, we used the content of Table II and randomly generated content of Table III. Figure 4 is the result of running the simulation for 100 times, showing that UFSA results in less hardship for the user.

Figure 4a shows that with increasing the operation sensitivity, UFSA results in less hardship. Greedy method causes higher hardship by just acting greedy. The Greedy method selects harder authentication methods if they have higher ratio of score over hardship. As a result, it ends up with higher hardship comparing to UFSA which uses a dynamic programming solution to select a solution with lower hardship. This figure shows that when operation sensitivity is 80, UFSA causes up to 18% less hardship compared to the Greedy method.



(a) Impact of operation sensitivity on the user hardship

(b) Impact of implicit factors on the user hardship

Fig. 4: Evaluating perceived user hardship in different working conditions.

In addition, our method shows up to 29% less hardship comparing to the Naïve method. The Naïve method selects a subset for authentication just by considering the hardship of the factors. It chooses factors with lower hardship but have lower authentication score over hardship ratio too. So, it ends up at a higher authentication hardship.

Figure 4b considers impact of the implicit weight on the accumulated hardship. It shows that when the implicit factors are not valid and hence the implicit weight of w becomes less, the user perceived hardship increased by requiring higher explicit authentication score. However, when a user's security condition improves by providing higher implicit factors to the learning component, the system adapts to the situation and requires less explicit authentication score resulting in less hardship for the users. It is shown in the Figure 4b that when implicit authentication weight increases, UFSA causes less inconvenience compared to the Greedy and Naïve methods.

VI. Conclusion and Future Work

Accessing cloud services through vulnerable devices, such as hand held computers, has become prevalent. Therefore, there is an increasing demand for secure authentication to access sensitive services hosted on clouds. Current multi-factor authentication methods cause user fatigue and are proven to be non-effective. We need authentication solutions that maintain ease-of-use in spite of frequent authentication processes. In this paper, we proposed an architecture based on progressive multi-factor authentication that uses a combination of explicit and implicit factors to provide the required security for accessing different levels of cloud services. In this architecture, a user needs to obtain more authentication score to be able to access more sensitive (i.e., higher level) cloud services. To grant access to a given level of cloud services, we proposed an algorithm that chooses a subset of authentication factors with the minimum perceived hardship for users. In addition, to increase security and user convenience, we consider the weight of implicit authentication factors to calculate the score that the user has to obtain using explicit authentication factors to be able to access a cloud service. Our simulation results showed that our proposed method reduces the users' perceived authentication hardship by up to 29% and 18% compared to the Naïve and Greedy authentication methods, respectively.

There are several interesting avenues of future work for this research. We are currently developing a meta-learning engine to provide an accurate multi-factor implicit authentication. We are also planning to extend the current work to determine the preferred authentication factors automatically and assign the authentication hardships without user involvement.

References

[1] C. Bo, L. Zhang, X.-Y. Li, Q. Huang, and Y. Wang. Silentsense: silent user identification via touch and movement behavioral biometrics. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 187–190. ACM, 2013.

[2] D. d. Borde. Selecting a two-factor authentication system. *Network Security*, 2007(7):17–20, 2007.

[3] C. Braz and J.-M. Robert. Security and usability: the case of the user authentication methods. In *Proceedings of the 18th International Conferenceof the Association Francophone d'Interaction Homme-Machine*, pages 199–203. ACM, 2006.

[4] M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 426(1871):233–271, 1989.

[5] S. Chaudhari, S. Tomar, and A. Rawat. Design, implementation and analysis of multi layer, multi factor authentication (mfa) setup for webmail access in multi trust networks. In *Emerging Trends in Networks and Computer Communications (ETNCC), 2011 International Conference on*, pages 27–32. IEEE, 2011.

[6] R. Chow, M. Jakobsson, R. Masuoka, J. Molina, Y. Niu, E. Shi, and Z. Song. Authentication in the clouds: a framework and its application to mobile users. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, pages 1–6. ACM, 2010.

[7] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Pro-*

ceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 987–996. ACM, 2012.

[8] P. A. Fahmi, E. Kodirov, D.-J. Choi, G.-S. Lee, A. Mohd Fikri Azli, and S. Sayeed. Implicit authentication based on ear shape biometrics using smartphone camera during a call. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pages 2272–2276. IEEE, 2012.

[9] T. Feng, Z. Liu, K.-A. Kwon, W. Shi, B. Carbunar, Y. Jiang, and N. Nguyen. Continuous mobile authentication using touchscreen gestures. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, pages 451–456. IEEE, 2012.

[10] T. Feng, J. Yang, Z. Yan, E. M. Tapia, and W. Shi. Tips: context-aware implicit user identification using touch screen in uncontrolled environments. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, page 9. ACM, 2014.

[11] R. Fischer and M.-Y. Kao. Multi-domain sandboxing: An overview. Technical report, Citeseer, 2000.

[12] H. Gascon, S. Uellenbeck, C. Wolf, and K. Rieck. Continuous authentication on mobile devices by analysis of typing motion behavior. In *Sicherheit*, pages 1–12, 2014.

[13] F. Gens. It cloud services user survey, pt. 2: Top benefits and challenges. *IDC eXchange*, 2008.

[14] L. Gong. A security risk of depending on synchronized clocks. *ACM SIGOPS Operating Systems Review*, 26(1):49–53, 1992.

[15] C. Herley. More is not the answer. 2014.

[16] P. Ho and J. Armington. A dual-factor authentication system featuring speaker verification and token technology. In *Audio-and Video-Based Biometric Person Authentication*, pages 128–136. Springer, 2003.

[17] H. Inoue and S. Forrest. Inferring java security policies through dynamic sandboxing. In *PLC*, pages 151–157, 2005.

[18] I. Ion. *User-centered security mechanisms for protecting information sharing in the cloud*. PhD thesis, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 20702, 2012, 2012.

[19] M. Jakobsson, E. Shi, P. Golle, and R. Chow. Implicit authentication for mobile devices. In *Proceedings of the 4th USENIX conference on Hot topics in security*, pages 9–9. USENIX Association, 2009.

[20] H. Khan, A. Atwater, and U. Hengartner. Itus: an implicit authentication framework for android. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 507–518. ACM, 2014.

[21] T. Khatiwala, R. Swaminathan, and V. Venkatakrishnan. Data sandboxing: A technique for enforcing confidentiality policies. In *Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual*, pages 223–234. IEEE, 2006.

[22] A. M.-H. Kuo. Opportunities and challenges of cloud computing to improve health care services. *Journal of medical Internet research*, 13(3), 2011.

[23] L. Li, X. Zhao, and G. Xue. Unobservable re-authentication for smartphones. In *NDSS*, 2013.

[24] M. Lori. Data security in the world of cloud computing. *Co-published by the IEEE Computer And reliability Societies*, pages 61–64, 2009.

[25] M. Maass. *A Theory for Applying Sandboxes Effectively*. PhD thesis, Carnegie Mellon University, 2014.

[26] R. E. Newman, P. Harsh, and P. Jayaraman. Security analysis of and proposal for image-based authentication. In *Security Technology, 2005. CCST'05. 39th Annual 2005 International Carnahan Conference on*, pages 141–144. IEEE, 2005.

[27] M. Radhakrishnan and J. A. Solworth. Quarantining untrusted entities: Dynamic sandboxing using leap. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 211–220. IEEE, 2007.

[28] T. V. N. Rao and K. Vedavathi. Authentication using mobile phone as a security token. *International Journal of Computer Science & Engineering Technology*, 1(9):569–574, 2011.

[29] O. Riva, C. Qin, K. Strauss, and D. Lymberopoulos. Progressive authentication: Deciding when to authenticate on mobile phones. In *USENIX Security Symposium*, pages 301–316, 2012.

[30] N. Sae-Bae, K. Ahmed, K. Isbister, and N. Memon. Biometric-rich gestures: a novel approach to authentication on multi-touch devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 977–986. ACM, 2012.

[31] M. A. Salehi, T. Caldwell, A. Fernandez, E. Mickiewicz, E. W. D. Rozier, S. Zonouz, and D. Redberg. RESeED: Regular Expression Search over Encrypted Data in the Cloud. In *Proceedings of the 7th IEEE International Conference on Cloud Computing*, CLOUD '14, pages 673–680, 2014.

[32] F. E. Sandnes and X. Zhang. User identification based on touch dynamics. In *Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2012 9th International Conference on*, pages 256–263. IEEE, 2012.

[33] E. Shi, Y. Niu, M. Jakobsson, and R. Chow. Implicit authentication through learning user behavior. In *Information Security*, pages 99–113. Springer, 2011.

[34] T. Shinagawa, K. Kono, and T. Masuda. Flexible and efficient sandboxing based on fine-grained protection domains. In *Software Security—Theories and Systems*, pages 172–184. Springer, 2003.

[35] M. Sun and G. Tan. Jvm-portable sandboxing of java's native libraries. In *Computer Security–ESORICS 2012*, pages 842–858. Springer, 2012.

[36] A. Verma and S. Kaushal. Cloud computing security issues and challenges: a survey. In *Advances in Computing and Communications*, pages 445–454. Springer, 2011.