MACA: A Privacy-Preserving Multi-factor Cloud Authentication System Utilizing Big Data

Wenyi Liu, A. Selcuk Uluagac, and Raheem Beyah
GT CAP Group, The School of ECE
Georgia Institute of Technology
Atlanta, GA 30332, USA
{wliu76@gatech.edu}@gatech.edu, {selcuk, rbeyah}@ece.gatech.edu

***Abstract*—Multi-factor authentication (MFA) is an approach to user validation that requires the presentation of two or more authentication factors. Given the popularity of cloud systems, MFA systems become vital in authenticating users. However, MFA approaches are highly intrusive and expose users' sensitive information to untrusted cloud servers that can keep physically identifying elements of users, long after the user ends the relationship with the cloud. To address these concerns in this work, we present a *privacy-preserving multi-factor authentication* system utilizing the features of big data called MACA. In MACA, the first factor is a password while the second factor is a hybrid profile of user behavior. The hybrid profile is based on users' integrated behavior, which includes both host-based characteristics and network flow-based features. MACA is the first MFA that considers both user privacy and usability combining big data features (26 total configurable features). Furthermore, we adopt fuzzy hashing and fully homomorphic encryption (FHE) to protect users' sensitive profiles and to handle the varying nature of the user profiles. We evaluate the performance of our proposed approach through experiments with several public datasets. Our results show that our proposed system can successfully validate legitimate users while detecting impostors.**

***Index Terms*—Authentication in Cloud, Fuzzy Hashing, Fully Homomorphic Encryption, Privacy-Preserving Authentication**

I. Introduction

There is great demand to establish a multi-factor authentication (MFA) system which requires two or more authentication factors (i.e., knowledge, possession, identity) to validate users during their login into a cloud service. Several popular services employ MFA, usually as an optional feature which is deactivated by default, such as Amazon [1], Google [2], Microsoft [3]. Each of these techniques requires an out-of-band channel (e.g., an App, text message) and an extra step for a user, which reduces usability. Similarly, there have been quite a few academic proposals for MFA systems [4]–[8]. In a typical MFA system, each user is verified via the first authentication factor (usually password) along with a second or even a third factor such as smartcards [9], fingerprints [10], or user's mouse movements [8]. MFA solutions based on physical devices or physiological characteristics depend on the introduction of specialized hardware, such as a token or a fingerprint reader, which hinders usability and deployability by causing additional cost for manufacturing and implementation. Alternatively, the more usable (and therefore more likely to be widely-adopted) MFA solutions are based on users' behavior; however, they do little to protect their privacy. *This is critical, given that the validity of the specific user characteristics shared with a cloud service will likely significantly outlast the period of time for which the services are needed [11].*

To address these challenges, in this paper, we propose a *privacy-preserving* multi-factor authentication system called MACA, which collects hybrid user behavior profiles to serve as a second authentication factor along with the user password as the first. MACA is the first MFA that considers both user privacy and usability. Instead of just focusing on one specific category of user behavior, like system processes or user's mouse movements, we integrate features from several categories to generate a user's profile. We also adopt fuzzy hashing [12] and fully homomorphic encryption (FHE) [13] techniques to ensure that a user's personal information is not leaked to cloud or a third party and that the system is able to tolerate variations in a user's plaintext and corresponding cryptographic profiles. Further, MACA incorporates big data features efficiently in the authentication process. Specifically, MACA uses a large combination of host-based characteristics and network-based features to profile users. Combining multiple features (total 26 configurable features) and the large amount of data associated with each feature enable a much simpler, threshold-based user classification (instead of expensive machine learning) on the cloud. Since the operations for user classification are simple, the numerical portion of the profile can be verified using FHE. For example, using FHE a cloud operator can ascertain whether a user's average throughput value is in an acceptable range without knowing the actual value. Finally, for the string portion of the user profile, fuzzy hashing is used to match similar strings without knowing the actual values of the strings.

We performed experiments using a user profile database derived from several public datasets [14]–[16] and a dataset we generated. Based on those data, we evaluate the performance of the proposed system in terms of *recall, false positive rate, size of information required for authentication, system overhead*, and *resource utilization*. Our results show that the proposed scheme shows significant promise for detecting impostors from legitimate users.

The remainder of this manuscript is organized as follows.

We review the related work in Section II. Assumptions and the adversary model are presented in Section III. We explain the details of MACA in Section IV. We present our experimental design and discuss the results in Section V. We conclude this paper in Section VI.

II. Related Work

Authentication via more than one factor, Multi-Factor Authentication (MFA), has become an increasingly essential component for cloud systems - a vital means to ensure that users, no matter where they are, are in fact who they claim to be and thus are authorized to gain access to cloud resources. A number of researchers have proposed the design and implementation of MFA systems [4]–[8], with each presenting its own specific advantages and tradeoffs. *Knowledge factor* (i.e., passwords) is the most ubiquitous authentication factor. It is widely-known that the sole use of passwords has many weaknesses. Nevertheless, passwords are still in use and are the de facto standard [17]. MFA systems using passwords along with a *possession factor* are commonly found in electronic commerce and online banking. Security tokens, also called One-Time-Password (OTP) tokens, is one of the most commonly-used *possession factor*, which generates a pseudo-random number at pre-determined intervals (e.g., RSA SecurID [18] and VeriSign Security Token [5]). Additionally, these cards serve as an extra authentication factor especially in corporate network environments. For instance, in [9], Kumar proposed a secure remote user authentication scheme with smart cards for corporate networks. Unfortunately, an MFA system with a *possession factor* usually depends on the distribution of some specific device, which is cumbersome and not user-friendly. Besides, the introduction of physical devices may pose further security risks if the devices are lost, stolen or replicated without the knowledge of the legitimate user. Finally, authentication via an *identity factor* is also a well-studied area of research. *Identity factors* are further categorized as either physiological biometrics or behavioral characteristics [6], [19]. Physiological biometrics, such as fingerprints, iris and face, have already drawn considerable attention in academia and have been implemented widely in industry [10]. Behavioral biometrics, such as mouse movements, keystroke dynamics, graphical passwords have also gained popularity in the research community [7]. Similar to MFA systems with *possession factors*, MFA systems with physiological biometrics suffer from a relatively low usability and deployability due to the implementation cost of biometrics recognition devices. To the best of our knowledge, no consideration has been given to the privacy issue when authenticating based on user behavior for cloud systems utilizing big data. This is critical, given that the validity of the specific user characteristics shared with a cloud operator will likely significantly outlast the period of time for which the cloud services are needed. Therefore, our goal in this work is to develop a *privacy-preserving* multi-factor authentication system based on passwords along with hybrid user profiles, that considers both usability and privacy issues. *As we consider a large amount of host-based characteristics and network-based features and the corresponding data, the large amount of data associated with each feature enable a much simpler, threshold-based user classification (instead of expensive machine learning) in the cloud.*

III. Assumptions and Adversary Model

In our system, we make the following two assumptions: 1) *Perfect knowledge assumption*: We assume that the adversary has perfect knowledge of the multi-factor authentication system including the strategy of user profile acquisition, the mechanism of profile encryption/hashing, and the details of the authentication protocol. 2) *First-Factor knowledge assumption*: We assume that the adversary knows the victim's first authentication factor, which is the user password.

In our adversary model, a malicious entity can attack the proposed multi-factor authentication system via impersonating a legitimate user (victim) in order to gain access to the victim's account. We specifically consider the following two types of adversaries: *brute-force attacker* and *honest-but-curious server attacker*.

1) *Brute-force attacker*: A computationally bounded third-party adversary may attempt to authenticate with a spoofed second authentication factor by exhaustively searching for the correct user profile. Such an attack consists of enumerating all possible user profiles until the correct one is found, which, in the worst case, would involve traversing the entire message space. 2) *Honest-but-curious server attacker*: In our system, we assume the server in the cloud that processes the authentication requests is *honest-but-curious* that (a) stores incoming cryptographic data without tampering with it; (b) honestly processes each authentication request and returns the corresponding outcome; (c) but tries to derive the underlying sensitive information from the user's cryptographic profile.

Moreover, we use a fuzzy hashing scheme implemented as `sdhash` [20] by Rousseu [21] to match data with similarities. In industry, fuzzy hashing has been applied in the realm of security forensics, especially in identifying morphing malware and spam. In our system, we adopt fuzzy hashing to evaluate the similarity of two fuzzy hash values of user behavior features in the forms of strings, without revealing the user's sensitive information to the authentication server.

Finally, we achieve the privacy of a user's profile with fully homomorphic encryption (FHE) [13], [22], [23]. Specifically, we used and improved the open-source library `Simple-Homomorphic-Encryption` [24] in our authentication system, which is a C++ implementation of homomorphic encryption discussed in [22]. The user sends the information encrypted with public key pk by function $Encrypt$ to the server. As listed in Equation 1, the encryption scheme ε has an algorithm $Evaluate_\varepsilon$ that, given plaintext $\pi_1, \pi_2, ...\pi_t$, for any valid ε, private, public key pair (sk, pk), any circuit C, and any ciphertext $\psi_i \longleftarrow Encrypt_\varepsilon(pk, \pi_i)$, yields

$$\begin{aligned} \psi &\longleftarrow Evaluate_\varepsilon(pk, C, \psi_1 ... \psi_t) \\ \text{such that } &Decrypt_\varepsilon(sk, \psi) = C(\pi_1, \pi_2 ... \pi_t) \end{aligned} \tag{1}$$

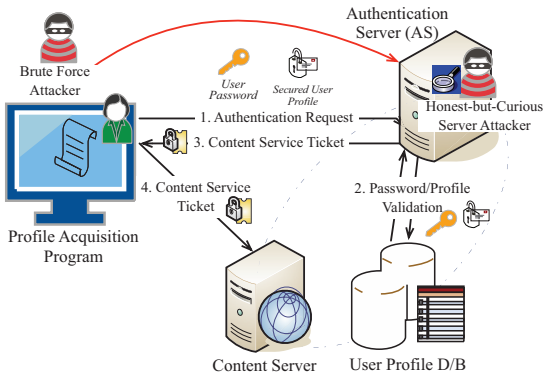

Fig. 1. Overview of the Privacy-Preserving MFA System

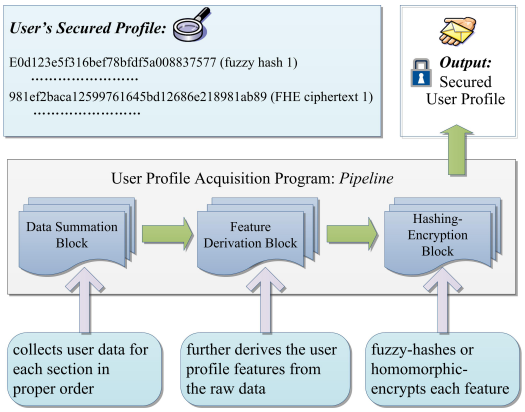

Fig. 2. The Pipeline of User Profile Acquisition Program

The server in the cloud does operations on the encrypted numbers by function $Evaluate$ with public key pk and outputs ψ. The server sends ψ back to the user. The user then decrypts ψ by function $Decrypt$ with his private key sk and obtains the result of $C(\pi_1, \pi_2 ... \pi_t)$. In this way, the server conducts the desired operation for the user without acquiring any plaintext.

IV. Proposed System

In this section, we introduce the design of MACA. Our proposed system collects the user behavior to serve as a second authentication factor along with the user's password. However, unlike conventional user behavior profiling, the user information acquisition, transmission and storage all occur in a privacy-preserving fashion.

A. System Overview

The architecture of the developed system has four primary components and is shown in Figure 1: (1) an open-source *profile acquisition program* (PAP) that runs on the user's local host; (2) a *user profile database* (UPDB) that stores the user's information in a privacy-preserving fashion; (3) an *authentication server* (AS) that processes and validates user's login request in the cloud environment; and (4) a content server.

The first time a user uses MACA with a specific site, he/she must go through the process of *enrollment*. During enrollment, the user information acquisition program collects a user profile, denoted by P and then hashes and encrypts P with the FHE public key pk. Then, the user ID and user-assigned password, denoted by uid and Psw, along with the cryptographic user profile, denoted by $\dot{P}$, are passed to the AS. The AS will interact with the UPDB and thus insert $\dot{P}$ into the user profile database. For each login attempt afterwards, the individual will pass his uid, typed password, denoted by Psw' as well as the newly captured user profile in cipher text, denoted by $\dot{P}'$. The AS is responsible for evaluating how much $\dot{P}'$ is different from $\dot{P}$ (hereinafter referred to as *distance*) and returning the corresponding authentication result denoted as $AuthResult$, a boolean value indicating authentication as *success* or *failure*. If $AuthResult$ is success, the AS will send a content service ticket along with $AuthResult$ to the user, which contains a session ID and a timestamp. Any user holding a valid service ticket may initiate a service request to the content sever.

The major challenge of the multi-factor authentication system is how to preserve the privacy of the user profile from servers and any third party while enabling the server to determine distance between user profiles. To achieve this, we use fuzzy hashing and fully homomorphic encryption techniques. From Equation 1, we can derive operations that are used by the server in the cloud on ciphertext values $(\psi_1..\psi_k)$ sent by the user when generating the ψ. Specifically, we implemented two simple arithmetic operations, *substraction* and *division* using the underlying And/Xor gates that were proven to be secure [23]. Assuming that we have FHE key pair (pk, sk), $\dot{a_H} \leftarrow FHE\_Encrypt\ (a, pk)$, $\dot{b_H} \leftarrow FHE\_Encrypt(b, pk)$, $\psi_1 \leftarrow FHE\_Sub(\dot{a_H}, \dot{b_H}, pk)$, and $\psi_2 \leftarrow FHE\_Div\ (\dot{a_H}, \dot{b_H}, pk)$, we can show the existence of Equations 2 and 3 as follows:

$$FHE\_Decrypt(\psi_1, sk) = a - b \tag{2}$$

$$FHE\_Decrypt(\psi_2, sk) = a/b \tag{3}$$

B. User Profile Acquisition

To collect user profiles, we developed two user profile acquisition programs in C# and Python for Windows and Linux OSs, respectively. The program has three main steps as illustrated by the cascading blocks in Figure 2: *data summation, feature derivation*, and *hashing-encryption*.

The *Data Summation* block is responsible for collecting the user information in a *sliding window* - collection for some user information occurs continuously and at the end of each sliding window period, the collected information is handed over to the Feature Derivation block and Data Summation starts again. *Feature Derivation* block receives the raw data from the previous block and extracts the required features. After each feature is ready for processing, the Feature Derivation block passes the data to the next block. *Hashing-Encryption* block is responsible for generating a cryptographic profile based on all of the available features via fuzzy hashing and fully

TABLE I
FEATURES USED FOR USER PROFILE MODELING

Category	Section	# Features	Type
Host-Based	1 File System and Registry	2	string
	2 Mouse Dynamics	3	number
	3 Keystroke Activity	2	number
	4 System Process	4	string
Network-Based	5 Browser Information	3	string
	6 Flow-Based Features	19	number

homomorphic encryption. The fuzzy-hashed user profile is denoted by $\dot{P_F}$ while the homomorphically encrypted user profile is denoted by $\dot{P_H}$. Thus, the block outputs a hybrid cryptographic user profile $\dot{P} \longleftarrow \{\dot{P_F}, \dot{P_H}\}$. In addition, the Feature Derivation Block continues to produce feature information at the end of every window period.

Furthermore, as illustrated in Table I, user characteristics or features are divided into two categories: host-based and network-based, which are further classified into several sub-categories as follows.

1) *File System and Registry*: This sub-category consists of fuzzy hashes of the file hierarchy at a critical path and a portion of the registry contents (for Windows systems). 2) *Mouse Dynamics*: User's mouse movements can be characterized via three fine-grained metrics: *direction, curvature distance*, and *curvature angle*. Nan Zheng et al. [8] proved that these three angle-based features are relatively unique from person to person and independent of the computing platforms and can therefore be used to distinguish legitimate users from intruders. To obtain a stable and representative sample, we use the average values of these three metrics in a time window as several features in the user's profile. 3) *Keystroke Activity*: User's keystroke activity is modeled via two features: the average key press-down time and the average time interval between key presses. In MACA, these metrics of keystroke activity are captured and derived as described in [25] while a user is typing his password in order to log into the the operating system. 4) *System Processes*: This sub-category is composed of fuzzy hashes of active system process names. 5) *Browser Information*: In this sub-category, we utilize the auto-fill information in browsers. We derive the fuzzy hash of personal information with attributes of "Email", "Username" and "Address". The significance of auto-fill information is that one tends to have same auto-fill value for those frequently-used attributes, in different browsers or hosts. 6) *Flow-Based Features*: We model users' general network behavior via 19 flow-based features. Note that it is commonly known that flow-based features indicate the category of network traffic (e.g., stream video, online chat) and thus serve a good reflection of user's network usage patterns. In our system, we use the average values of each feature given a specific window of time. The process of generating a cryptographic user profile is given in Algorithm 1. Note that the user profiling approach we propose is an extensible and configurable framework, which means that one can always edit the user profile by deleting or inserting new user behavioral features.

TABLE II
DATA SOURCE FOR EXPERIMENTS

Section	Data Set	Subjects
Mouse Dynamic	NSKEYLAB Dataset [15]	10
Keystroke Activity	CMU Dataset [16]	51
General Network	DACS Dataset [14]	132

C. Authentication in the Cloud

To initiate an authentication attempt, the user passes the uid, Psw' for the first-factor authentication. The failure of first-factor authentication terminates the conversation. If the user passes the first step of the authentication, he then passes his newly generated user profile, $\dot{P}'$, to the server in the cloud. After evaluating the distance between $\dot{P}'$ and $\dot{P}$, the server returns a boolean value $AuthResult$, indicating the success or failure of the second-factor authentication. In order to yield $AuthResult$, we introduce a new concept, the *Accepted Distance Value (ADV)*. To define ADV, we first define $distance\ value$. Assuming P_{H_j} and $\dot{P}_{H_j}$ denote the values of the j^{th} feature in profile P and profile $\dot{P}$, the distance value between P_{H_j} and $\dot{P}_{H_j}$ is defined in Equation 4:

$$distance_j = \frac{|\dot{P}_{H_j} - P_{H_j}|}{P_{H_i}} \times 100 \tag{4}$$

Algorithm 1 *GenerateCryptographicUserProfile()*

Input: P_F, P_H, pk
Output: $\dot{P} \longleftarrow \{\dot{P_F}, \dot{P_H}\}$
1: **for** each $i \in [1, n]$ **do**
2: $\quad p_{\dot{F}_i} \longleftarrow FuzzyHash(p_{F_i})$;
3: **end for**
4: **for** each $j \in [1, m]$ **do**
5: $\quad p_{\dot{H}_j} \longleftarrow FHE\_Encrypt(p_{H_j}, pk)$;
6: **end for**
7: $\dot{P_F} \longleftarrow \{p_{\dot{F}_1} ... p_{\dot{F}_i} ... p_{\dot{F}_n}\}$;
8: $\dot{P_H} \longleftarrow \{p_{\dot{H}_1} ... p_{\dot{F}_j} ... p_{\dot{F}_m}\}$;
9: $\dot{P} \longleftarrow \{\dot{P_F}, \dot{P_H}\}$;
10: **return** $\dot{P}$;

V. EXPERIMENTATION AND EVALUATION

In this section, we evaluate the feasibility of our proposed system through a series of experiments on real data.

A. Experimental Setup

We conducted experiments with a combination of four datasets. The names of the public datasets used in our experiments are shown in Table II. 1) *NSKEYLAB Dataset* [15] is a dataset containing mouse dynamics information from 10 subjects, each of who accomplishes at least 30 data sessions. Each session consists of about 30 minutes of a user's mouse activity in a free environment [15], [26]. We derived three angle-based metrics using the approach proposed in [8], thus,

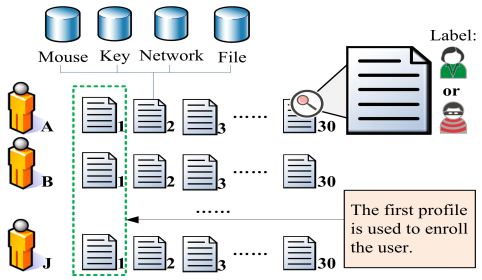

Fig. 3. The Generation of Hybrid User Profile Samples

generating three data points that represent the user's mouse movement profile. 2) *CMU Dataset* [16] is a dataset consisting of keystroke-timing information from 51 subjects (typists), each typing a password 400 times [16], [25]. 3) *DACS Dataset* [14] is a dataset consisting of the network trace for an educational organization. We used SplitPcap [27] to obtain 30-min-long separate pcap files and for each small pcap file, we ran TSTAT [28] to derive the flow-based features, thus, generating the user's network behavior profile. This data set spans two months. In addition to these three datasets, we generated the dataset that contains *file system* and *registry* information with 130 different software names (30 O/S pre-installed, 100 personalized) for 15 subjects. Then, based on these four datasets, we generated 300 different 30-min-long hybrid user profile samples.

Moreover, for each user, we registered his/her first piece of cryptographic hybrid user profile in the database (the enrollment of that user). Then, we appended a label to each profile indicating the owner of that profile. We randomly chose half of all the hybrid user profile samples and intentionally labeled them incorrectly making the actual owner of that profile appear as an intruder. For the other half of the hybrid user profile samples, we correctly labeled them, treating the owner as a legitimate user. Finally, we randomly split all the labeled user profiles into two equal sized parts: the training set and the testing set. We hashed and encrypted each profile in the testing set via Fuzzy Hashing and FHE as explained in Sections III, IV. We conducted all the experiments on a standard laptop computer with Intel CPU i5-M430 (2.27GHz) and RAM of 4 GB.

B. Results and Discussion

We evaluated our system in terms of recall, false positive rate, system overhead, size of authentication packet, and resource utilization:

1) *Recall*: Recall is the proportion of positive cases that are correctly identified and was calculated using $\frac{TP}{TP+FN}$. TP denotes True Positive and FN denotes False Negative while in our authentication system an intruder is labeled as *Positive* and a legitimate user as *Negative*. 2) *False Positive Rate (FPR)*: FPR is the proportion of negatives cases that are incorrectly classified as positive and was calculated using $\frac{TN}{FP+TN}$. TN denotes True Negative while FP denotes False Positive. 3) Size of Authentication Packets (hereafter referred to as *packet size*): Packet size is the size of all the authentication information transmitted from client to the server. 4) *System Overhead*: System Overhead is the initial latency during the entire multi-factor authentication process. We evaluated the overhead introduced by user profile acquisition, cryptography, data transmission, and server processing. 5) *Resource Utilization*: Resource Utilization is defined as how much system resources it takes for user information acquisition program to continuously retrieve information and derive features in terms of CPU and RAM utilization.

We conducted a series of experiments on the testing set containing 150 labeled hybrid user profile samples within different time window sizes for data collection. To decide whether a hybrid user profile is legitimate, we first determined appropriate thresholds for individual characteristics that comprise the profiles. Next, we used a majority vote to make the final decision on the legitimacy of a profile sample. A training set containing 150 labeled hybrid user profile samples served as a priori knowledge. The experimental results in terms of recall and false positive rate (FPR) are presented in Table III.

TABLE III
EXPERIMENT RESULTS: RECALL AND FPR

Time For Data Collection	Recall (%)	FPR (%)
5 min	74.2	26.9
10 min	75.4	19.7
20 min	78.8	16.2
30 min	80.8	14.7

From the table, it can be seen that the longer the time window for the data collection, the better the system performance is in terms of recall and FPR. When the time window is as long as 30 minutes (initial bootstrap latency - this drops to 0 for every attempt after 30 minutes), we achieve an optimal result with recall of 80.8% and FPR of 14.7%. Nevertheless, a longer data collection time tends to reduce the usability of the system. Therefore, there is a trade off between system accuracy and efficiency.

Next, we evaluated the other three system performance indicators: packet size, system overhead, and resource utilization. In terms of packet size, for the first-time user enrollment, the packet size is the sum of the size of the user's public key pk, uid, Password Psw, and cryptographic user profile $\dot{P}$. Since the size of uid and Psw are far less than the size of the rest, we can neglect uid and Psw in the calculations. Packet size, $Size$, was calculated using Equation 5, in which $pkSize$ denotes the size of public key pk, n denotes the number of string features, α denotes the size of each fuzzy hash, m denotes the number of number features and β denotes the size of each FHE ciphertext. In MACA, with $pkSize = 0.98MB, n = 9, \alpha = 0.125KB, m = 24$, and $\beta = 146KB$, the packet size for the enrollment is approximately 4507 KB. In the authentication procedure, the user is not expected to transmit their pk again; so, the packet size is around 3504 KB after enrollment.

$$Size = pkSize + n * \alpha + m * \beta \tag{5}$$

System overhead, T, includes the time spent on authentication

data transmission, T_t and the server process, T_s. We can derive T using Equation 6. T_t largely depends on the network environment. In a high-speed Internet environment, T_t is normally under 30 seconds. T_s was less than 5 seconds in our experiments. In total, the system latency after users initiate a login request is around 35 seconds. Note that system overhead may vary because it depends on the computing ability of both the client and the cloud and also to the specific network conditions.

$$T_{after} = T_t + T_s \tag{6}$$

In terms of resource utilization, in our experiments, the running of the user profile acquisition program takes less than 3% of CPU resources and about 15MB of RAM. The biggest proportion of computing resource consumption stems from the capturing of network packets. Because we only capture packet headers, the resource demanded for network monitoring is still in an acceptable range. As the statistics for resource utilization is based on a laptop computer with Intel CPU i5-M430 (2.27GHz) and a RAM of 4 GB, the CPU utilization percentage will decrease with a more powerful computer.

The evaluation of recall, FPR, packet size, system overhead, and resource utilization demonstrates the feasibility of MACA. Note that as both packet size and overhead are largely dependent on the time and space complexity of FHE; with further improvements in FHE, one can reduce both packet size and system overhead, thereby, further boosting performance.

VI. Conclusion

In this work, we developed a privacy-preserving multi-factor authentication system without introduction of any extra physical device for cloud systems utilizing big data features. In our system, called MACA, the first factor is a password and the second factor is a hybrid user profile that summarizes user behavior. MACA focuses on the privacy preservation of the second factor, which has two advantages over previously proposed systems. First, user privacy is not leaked to ubiquitous cloud computing environment with FHE and fuzzy hashing. Second, the hybrid user profiling model is highly usable and configurable and integrates a lot of features and corresponding data, which enables simple privacy-preserving MFA operations with FHE and fuzzy-hashing calculations. One can always modify the feature list (total 26 configurable features) for user profiling in MACA according to the actual circumstances. We evaluated the system performance via a series of experiments utilizing four different datasets, resulting in an optimal recall of 80.8%and FPR of 14.7%. Also, both system overhead and resource utilization are within the acceptable range, which substantiates the feasibility of our scheme. We plan to extend our work by adding more features and including a weighting scheme on features that can be configured by the system administrator and plan to improve performance.

References

[1] "Awe multi-factor authentication frequently-asked questions," 2012, http://aws.amazon.com/cn/mfa/faqs/.

[2] "About two-step verification," 2012, http://support.google.com/accounts/bin/topic.py?topic=28786.

[3] "Two factor authentication: What is a microsoft account security code?" 2012, http://answers.microsoft.com/en-us/windowslive/ forum/liveid-wlsecurity/two-factor-authentication-tfa-what-is-a-microsoft/.

[4] C. Paul, E. Morse, A. Zhang, Y.-Y. Choong, and M. Theofanos, "A field study of user behavior and perceptions in smartcard authentication," in *Human-Computer Interaction INTERACT*, ser. Lecture Notes in Computer Science. Springer, 2011, vol. 6949, pp. 1–17.

[5] "Validation and id protection," 2012, https://idprotect.verisign.com/mainmenu.v.

[6] M. Sujithra and G. Padmavathi, "Next generation biometric security system: an approach for mobile device security," in *Proc. of the 2nd ACM International Conference on Computational Science, Engineering and Information Technology*, 2012, pp. 377–381.

[7] Z. Jorgensen and T. Yu, "On mouse dynamics as a behavioral biometric for authentication," in *Proc. of the 6th ACM Symposium on Information, Computer and Communications Security*, 2011, pp. 476–482.

[8] N. Zheng, A. Paloski, and H. Wang, "An efficient user verification system via mouse movements," in *Proc. of the 18th ACM conference on Computer and communications security*, 2011, pp. 139–150.

[9] M. Kumar, "New remote user authentication scheme using smart cards," *Consumer Electronics, IEEE Transactions on*, vol. 50, no. 2, pp. 597–600, 2004.

[10] A. Bhargav-Spantzel, A. Squicciarini, and E. Bertino, "Privacy preserving multi-factor authentication with biometrics," in *Proc. of the 2nd ACM workshop on Digital identity management*, 2006, pp. 63–72.

[11] "Apple finally reveals how long siri keeps your data," April 2013, http://www.wired.com/wiredenterprise/2013/04/ siri-two-years/.

[12] J. Kornblum, "Identifying almost identical files using context triggered piecewise hashing," *Digital Investigation*, vol. 3, Supplement, no. 0, pp. 91 – 97, 2006, the Proc. of the 6th Annual Digital Forensic Research Workshop.

[13] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.

[14] "Pcap trace: Trace 6, packet headers," 2012, http://traces.simpleweb.org.

[15] "Mouse data for continuous authentication," April 2012, http://nskeylab.xjtu.edu.cn/people/cshen/.

[16] "Keystroke dynamics: Benchmark data set," 2012, http://www.cs.cmu.edu/~keystroke.

[17] J. Bonneau, C. Herley, P. C. v. Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *Proc. of the IEEE Symposium on Security and Privacy*, 2012, pp. 553–567.

[18] "Rsa securid - world's leading two-factor authentication," 2012, http://www.emc.com/security/rsa-securid.htm.

[19] R. V. Yampolskiy and V. Govindaraju, "Behavioural biometrics: a survey and classification," *Int. J. Biometrics*, vol. 1, no. 1, pp. 81–113, Jun. 2008.

[20] "sdhash," 2012, http://roussev.net/sdhash/sdhash.html.

[21] V. Roussev, "Data fingerprinting with similarity digests," in *Advances in Digital Forensics VI*. Springer, 2010, vol. 337, pp. 207–226.

[22] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," Cryptology ePrint Archive, Report 2009/616, 2009, http://eprint.iacr.org/.

[23] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully homomorphic encryption over the integers with shorter public keys," in *Advances in Cryptology CRYPTO 2011*, ser. Lecture Notes in Computer Science, P. Rogaway, Ed. Springer, 2011, vol. 6841, pp. 487–504.

[24] S. Crane, "Simple-homomorphic-encryption," 2012, https://github.com/rinon/ Simple-Homomorphic-Encryption.

[25] K. Killourhy and R. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *Dependable Systems Networks, 2009. DSN '09. IEEE/IFIP International Conference on*, 2009, pp. 125–134.

[26] C. Shen, Z. Cai, and X. Guan, "Continuous authentication for mouse dynamics: A pattern-growth approach," in *42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2012, pp. 1–12.

[27] "Netresec splitcap - a fast pcap file splitter," 2012, http://www.netresec.com/?page=SplitCap.

[28] "Tstat - tcp statistic and analysis tool," 2012, http://tstat.tlc.polito.it/index.shtml.