Received: 12 June 2019 | Revised: 18 July 2019 | Accepted: 31 July 2019 | Published on: 27 August 2019

DOI: 10.1002/spy2.88

ORIGINAL ARTICLE

WILEY

A privacy-preserving multifactor authentication system

Abbas Acar[1] | **Wenyi Liu**[2] | **Raheem Beyah**[2] | **Kemal Akkaya**[1] | **Arif Selcuk Uluagac**[1]

[1]School of Electrical and Computer Engineering, Florida International University, Miami, Florida

[2]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia

Correspondence
Abbas Acar, School of Electrical and Computer Engineering, Florida International University, Miami, FL, 33172.
Email: aacar001@fiu.edu

Funding information
National Science Foundation, NSF-CAREER-CNS-1453647, NSF-CNS-1718116

Abstract

In recent years, there has been a significant number of works on the development of multifactor authentication (MFA) systems. Traditionally, behavioral biometrics (eg, keystroke dynamics) have been known to have the best usability because they do not require one to know or possess anything—they simply communicate "how you type" to an authenticator. However, though highly usable, MFA approaches that are based on biometrics are highly intrusive, and users' sensitive information is exposed to untrusted servers. To address this privacy concern, in this paper, we present a privacy-preserving MFA system for computer users, called PINTA. In PINTA, the second factor is a hybrid behavioral profile user, while the first authentication factor is a password. The hybrid profile of the user includes host-based and network flow-based features. Since the features include users' sensitive information, it needs to be protected from untrusted parties. To protect users' sensitive profiles and to handle the varying nature of the user profiles, we adopt two cryptographic methods: Fuzzy hashing and fully homomorphic encryption (FHE). Our results show that PINTA can successfully validate legitimate users and detect impostors. Although the results are promising, the trade-off for privacy preservation is a slight reduction in performance compared with traditional identity-based MFA techniques.

KEYWORDS

fuzzy hashing, homomorphic encryption, multifactor authentication, privacy-preserving

1 | INTRODUCTION

Both for highly sensitive systems such as online retail and e-banking and relatively less critical systems such as desktop machines in a corporate Intranet and social networks, it is crucial to protect users' accounts and assets from malicious third parties. Only password-based authentication systems suffer from many weaknesses, like password cracking, susceptibility to phishing and cross-site password reuse. Once the password is compromised, an adversary can easily misuse the victim's account. Thus, there is a great demand to establish a multifactor authentication (MFA) system, which requires two or more authentication factors (ie, knowledge, possession, identity) to validate users during their login.

Popular web services such as Amazon Web Services,[1] Google Accounts,[2] and Microsoft Outlook[3] have already deployed MFA. However, in all of these techniques, an out-of-band channel (eg, an App, text message) and an additional action from the user is required. This reduces usability significantly. Similarly, in the literature, there have been a number of academic works proposing MFA systems.[4–8]

Abbas Acar and Wenyi Liu contributed equally to this work.

The more practical (and therefore more likely to be widely adopted) MFA solutions are based on users' behavior; however, they do little to protect their privacy. In particular, in an MFA system, users face the risk of exposing their personal information to database servers or a malicious adversary. First, the owner of the database server may use it for malicious purposes (eg, selling user's information for economic interest) or if an adversary breaks into the database server and succeeds in obtaining the profile belonging to the targeted user, he or she can masquerade as a legitimate user by crafting required authentication factors.

Nonetheless, there are several challenges in achieving a privacy-preserving MFA system based on user profiles. First, a widely acknowledged challenge in the area of user profiling is how to accurately model a user's behavior while it constantly changes.[9] Note that even for the same user, there may be a difference between two profiles collected at different times. The second challenge is to identify a unique user from others based on their own varying profiles. The third challenge is to enable a server to verify the aforementioned profile, given that: (a) it cannot be read by the server (to preserve user privacy), and (b) it will vary over time (so standard cryptography cannot be used).

In this paper, we propose a privacy-preserving method for MFA systems, called PINTA, in which the privacy of the collected hybrid user behavior profiles serving as a second authentication factor is protected from the authentication server (AS). Moreover, we also adopt fuzzy hashing[10] and fully homomorphic encryption (FHE)[11] techniques to ensure that a user's personal information is not leaked to servers or a third party. Furthermore, PINTA uses a large combination of host-based characteristics and network-based features to profile users. The combination of multiple features (26 configurable features in total) enables a much simpler, distance metric-based user classification instead of expensive machine learning. Finally, while fuzzy hashing is used to match the strings portion of the user profile, FHE is used to match integer numbers without knowing the actual value.

For the experiments, we used a user profile database (UPDB) derived from several public datasets[12–14] and a dataset we generate. Our results show that the proposed scheme can well detect imposters from legitimate users while protecting user privacy. The contributions of this work are summarized as follows:

- We introduce a user profiling approach based on users' hybrid behavior, which includes both host-based characteristics and network-based features.
- Our approach provides an extensible and configurable framework combining 26 different features; thus, the user profile template (ie, host-based characteristics and network-based features) can be modified as needed.
- To protect user privacy, we adopt fuzzy hashing and FHE techniques for PINTA, which enables PINTA to handle differing cryptographic profiles belonging to the same individual.
- We implemented our authentication system and evaluated the performance of the proposed scheme via extensive experiments.

The remainder of this manuscript is organized as follows. We review the related work in the areas of both user profiling and MFA in Section 2. The problem formulation and preliminaries are presented in Section 3. We explain the details of our MFA system in Section 4. We present our experimental design, discuss the results, and analyze the security of the proposed system in Section 5. We conclude this paper in Section 6.

2 | RELATED WORK

In this section, we examine the related work in user profiling and MFA systems.

2.1 | Multifactor authentication

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* (ie, 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.[15] Thus, to reduce the security risk of the sole use of a *knowledge factor*, researchers have added *possession factor* and *identity factor* to authentication systems.

MFA systems using passwords along with a *possession factor* are commonly found in electronic commerce and online banking. Security token, also called one-time-password (OTP) token, is one of the most commonly-used *possession factor*, which generates a pseudo-random number at predetermined intervals (eg, RSA SecurID[16] and VeriSign Security Token[5]). Additionally, these cards serve as an additional authentication factor, especially in corporate network environments. For instance, in Reference 17, 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

TABLE 1 Comparative evaluation of PINTA

	Security	Privacy	Usability	Low Deployment Cost
Password-only	◐	na	●	◐
2FA	◐		◐	●
Czeskis et al.[18]	◐	◐		●
Sujithra et al.[6]	●			
Bhargav-Spantzel et al.[20]	●	●		
PINTA (this work)[a]	●	●	●	●

● = offers the benefit; ◐ = almost offers the benefit; no circle = does not offer the benefit.

[a]This paper is an extended version of an earlier version of this paper,[24] where the main proposed framework is the same with several changes. These changes include the following major improvements. First of all, a new implementation of underlying cryptographic primitives, as well as new weight adjustment technique, which improves the accuracy of the first paper. Moreover, security analysis has been performed.

replicated without the knowledge of the legitimate user. Czeskis et al.[18] first consider the usability of an MFA system with a *possession factor* by proposing authentication through opportunistic cryptographic identity. Nevertheless, their proposed scheme requires the presence of the user's phone, which limits the usability of the system.

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.[20] Behavioral biometrics, such as mouse movements, keystroke dynamics,[21–23] graphical passwords, though not widely utilized, have also gained popularity in the research community.[7] Similar to MFA systems with *possession factors*, MFA systems with physiological biometrics suffer from relatively low usability and deployability due to the implementation cost of biometrics recognition devices. Meanwhile, the downside of using behavioral characteristics is that the system may induce relatively low authentication accuracy and large system overhead.[8] To the best of our knowledge, no consideration has been given to the privacy issue when authenticating based on user behavior. This is critical, given that the validity of the specific user characteristics shared with a site will likely significantly outlast the period of time for which the site's services are needed. That is, the shared characteristic is not a mere pseudonym, but a characteristic that can identify a user for years to come.

Therefore, our goal in this work is to develop a *privacy-preserving* MFA system based on passwords along with hybrid user profiles that considers usability, privacy, and deployment cost. In Table 1, we perform a comparative evaluation of our proposed scheme in this work, PINTA, where we compare PINTA with its alternatives in terms of the benefits offered by the schemes. As can be seen from the Table 1, PINTA offers more benefits than its alternatives.

3 | PROBLEM FORMULATION AND PRELIMINARIES

In this section, we first introduce our assumptions and the adversary model. Then, we outline the design goals of our work. We also provide a brief introduction of *fuzzy hashing* and *homomorphic encryption*.

3.1 | Assumptions and adversary model

In our system, we make the following two assumptions:

Perfect knowledge assumption: We assume that the adversary has perfect knowledge of the MFA system including the strategy of user profile acquisition, the mechanism of profile encryption/hashing, and the details of the authentication protocol.

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 MFA system via impersonating a legitimate user (victim) in order to gain access to the victim's account. We note that we do not consider any low-level communication adversary model such as man-in-the-middle and replay attacks because those attacks can be prevented by the appropriate implementation of one or several security protocols[25] and, hence, such attacks are not within the scope of our work. We specifically consider the following two types of adversaries: *brute-force attacker* and *honest-but-curious server attacker*.

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.

Honest-but-curious server attacker: In our system, we assume the server 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.

3.2 | Design goals

The design goals of our work are outlined below.

Privacy preservation assurance: The system should guarantee that the privacy of each user is well preserved. To be specific, never should anyone including the *honest-but-curious server* and the malicious third party obtain the user profiles in plaintext. We analyze the security of our proposed system in Section 5.5. We adopt fuzzy hashing and FHE techniques to provide this assurance.

Authentication usability assurance: The system should ensure a pleasant user experience by satisfying the following three conditions. First, a login should not need extra effort other than typing the username-password pair. Second, the system overhead (ie, authentication delay) of the entire authentication process should be tolerable. Third, the program that runs on the client's machine should not consume a lot of computing resources. We evaluate system overhead and resource utilization in Section 5.

Authentication accuracy assurance: The system should ensure that the authentication is accurate with acceptable recall and false positive rate (FPR), which poses two challenges. First, the system should be adaptive and flexible enough to tolerate slight changes in the users' profiles. Second, the system should be sensitive enough to recognize a login request initiated by an adversary.

3.3 | Fuzzy hashing and homomorphic encryption

3.3.1 | Fuzzy hashing

The system proposed in this work uses fuzzy hashing, also called context-triggered piecewise hashing,[10] which is a hashing function that can match data with similarities. Jesse Kornblum first proposed a generic fuzzy hashing scheme in Reference 10 and implemented his algorithm as ssdeep.[26] In Rousseu's approach, each similarity digest *SD* for a byte stream is generated by employing a sequence of Bloom Filters, which are bit vectors used for space-efficient set representation. Given two similarity digests SD_1 and SD_2, the similarity digest score is generated by the function SD_{score} (SD_1, SD_2), which yields a score of zero for a mismatch or a matching score ranging from 1 to 100. In industry, fuzzy hashing has been applied in the realm of security forensics, especially in identifying morphing malware and spam.[27] 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 AS.

3.3.2 | Homomorphic encryption

The second technique, in this work, to achieve the privacy of a user's profile involves the application of FHE. After being an open problem for a long time, the first plausible FHE scheme was introduced by Craig Gentry in his PhD thesis[11] in 2009. Homomorphic encryption has an additional algorithm compared to the traditional encryption algorithms, which is called homomorphic evaluation algorithms and allows to be able to perform operations on the encrypted data without decrypting it. Particularly, an homomorphic encryption scheme ε has an algorithm $Evaluate_\varepsilon$ that, given plaintext $\pi_1, \pi_2, \ldots \pi_t$, for any valid ε, private, public key pair (sk, pk), any circuit C, and any ciphertext $\psi_i \leftarrow Encrypt_\varepsilon(pk, \pi_i)$, yields

$$\psi \leftarrow Evaluate_\varepsilon(pk, C, \psi_1 \ \ldots \ \psi_t)$$
$$\text{such that } Decrypt_\varepsilon(sk, \psi) = C(\pi_1, \pi_2 \ \ldots \ \pi_t). \tag{1}$$

A typical scenario of FHE is illustrated in Figure 1. In this scenario, the user encrypts the data with public key pk and the function *Encrypt* and sends the encrypted data to the server. The server in the cloud performs operations on the encrypted data

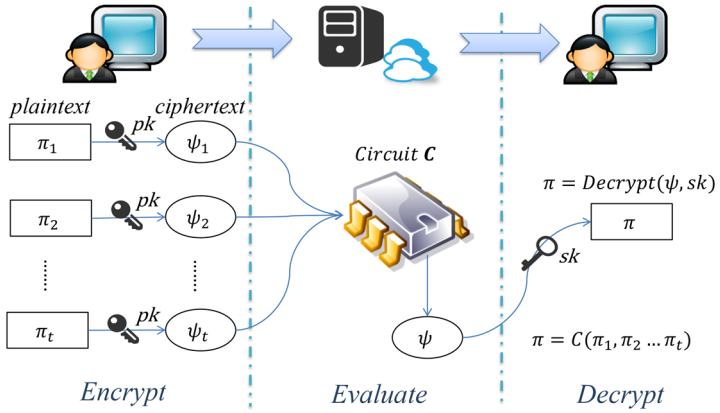

FIGURE 1 Fully homomorphic encryption

by using the 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 $C(\pi_1, \pi_2 \dots \pi_t)$. In this way, the server conducts the desired operation for the user without acquiring any plaintext. Van Dijk et al. later used Gentry's technique to establish a FHE system over integers.[28] Coron and his colleagues further improved van Dijk's work by reducing the size of public keys and time complexity.[29] Since then many improvements have been proposed[30–33] and the research on FHE still an active research area.[34] In our experiments, we used Microsoft's open-source homomorphic encryption library called SEAL[35] in our authentication system, which is a C++ implementation of homomorphic encryption. The current version of the SEAL implements two different encryption schemes: BFV and CKKS. We used the BFV version, which is originally proposed in Reference 36 and we chose the parameters to provide 128 bits security.

4 | PROPOSED SYSTEM

In this section, we introduce the design of PINTA. 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. Furthermore, to prove that none of these stages in the system violates user privacy, the proposed system is assumed to be open to the public, which corresponds to our *perfect knowledge assumption* as stated in Section 3.1.

4.1 | System overview

The architecture of the developed system has four primary components and is shown in Figure 2: (a) an open-source *profile acquisition program* (PAP) that runs on the user's local host; (b) a UPDB that stores the user's information in a privacy-preserving fashion; (c) an AS that processes and validates user's login request; and (d) a content server. We assume UPDB, AS, and the server are owned by the same organization, but this is not a strict requirement.

The first time a user uses PINTA with a specific site, he or she must go through the process of *enrollment*. During the enrollment, the user information acquisition program collects a user profile, denoted by P and then hashes or 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 UPDB. For each login attempt afterward, the individual will pass his *uid*, typed password, denoted by Psw' as well as the newly captured user profile in ciphertext, 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 a 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 server.

The major challenge of the privacy-preserving MFA system is how to preserve the privacy of the user profile from servers and any third party while enabling the server to determine the distance between user profiles. To achieve this, we use fuzzy hashing and FHE techniques. The seven operation primitives used in PINTA are summarized in Table 2.

From Equation (1), we can derive operations (Circuit C in Figure 1) that are used by the server on ciphertext values $(\psi_1 \dots \psi_k)$ sent by the user when generating the ψ. Specifically, we implemented two simple arithmetic operations, *subtraction* and *division* using the underlying And/Xor gates that were proven to be secure.[29] Assuming that we have FHE key pair (*pk*, *sk*), $a_H \leftarrow$

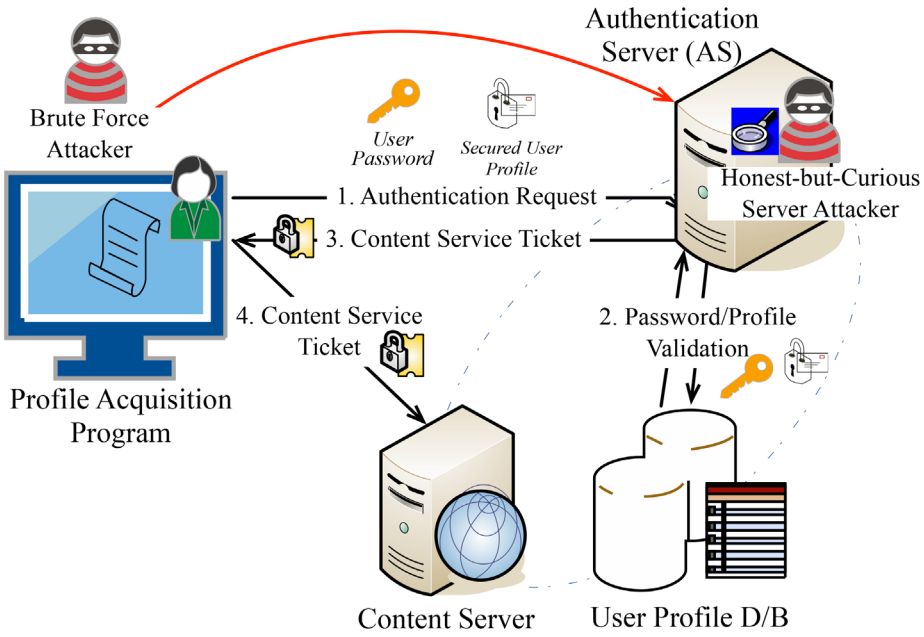

FIGURE 2 Overview of the privacy-preserving multifactor authentication system

TABLE 2 Operations in PINTA

Function	Operation
$FuzzyHash(a)$	Obtain the fuzzy hash of bytestream a, denoted by $\dot{a_F}$
$FuzzyCmp(\dot{a_F}\dot{b_F})$	Compare the distance between two fuzzy hash $\dot{a_F}$ and $\dot{b_F}$, return a value ranging from 0 to 100
$FHE\_KeyGen()$	Generate a key pair (pk, sk) for fully homomorphic encryption.
$FHE\_Encrypt(a, pk)$	Encrypt a with public key pk via FHE, outputting $\dot{a_H}$
$FHE\_Decrypt(\dot{a_H}, sk)$	Decrypt $\dot{a_H}$ with public key sk via FHE, output a
$FHE\_Sub(\dot{a_H}\dot{b_H}, pk)$	Subtract $\dot{a_H}$ with $\dot{b_H}$ via FHE under public key pk
$FHE\_Div(\dot{a_H}\dot{b_H}, pk)$	Divide $\dot{a_H}$ by $\dot{b_H}$ via FHE under public key pk

$FHE\_Encrypt\ (a, pk)$, $\dot{b_H} \leftarrow FHE\_Encrypt(b, pk)$, $\psi_1 \leftarrow FHE\_Sub(\dot{a_H}, \dot{b_H}, pk)$, 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:

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

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

4.2 | User profile acquisition

To collect user profiles, we developed two user PAPs in C# and Python for Windows and Linux OSs, respectively.

4.2.1 | Hybrid user profiling model

The program has three main steps as illustrated by the cascading blocks in Figure 3: *data summation*, *feature derivation*, and *hashing-encryption*.

Data summation: 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. To minimize the development effort, we use several third-party tools (ie, TSTAT[37]) to assist with data collection.

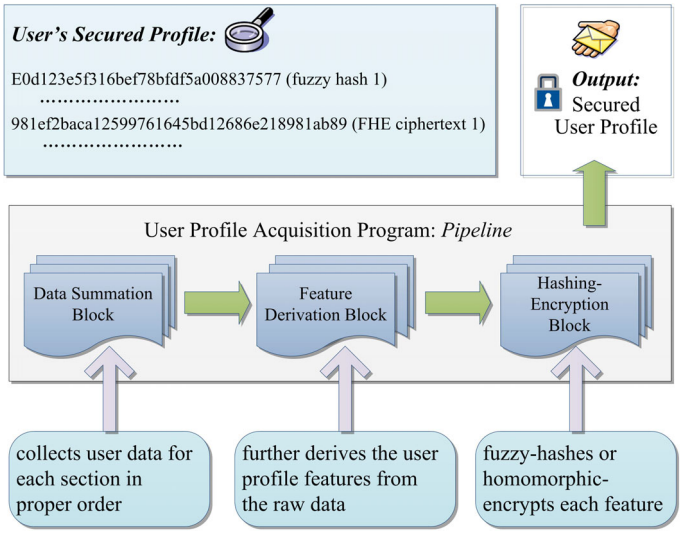

FIGURE 3 The pipeline of user profile acquisition program

TABLE 3 Features used for user profile modeling

Category	Section	Number[a]	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

[a]Number of features in the section.

TABLE 4 Notations used

$\dot{P}$	Cryptographic user profile stored	$P_F \leftarrow \{p_{F_1}, \ldots p_{F_i} \ldots p_{F_n}\}$	User's string features
$\dot{P}'$	Newly captured cryptographic user profile	$P_H \leftarrow \{p_{H_1}, \ldots p_{H_i} \ldots p_{H_m}\}$	User's number features
n	Number of string characteristics	$W_F \leftarrow \{w_{F_1}, \ldots w_{F_i} \ldots w_{F_n}\}$	Weight for each string feature
m	Number of number characteristics	$W_H \leftarrow \{w_{H_1}, \ldots w_{H_i} \ldots w_{H_m}\}$	Weight for each number feature
pk	User's public key	$k_j \leftarrow \{0, 1 \ldots k_j\}$	Threshold for the jth number feature
sk	User's private key	$V_j \leftarrow \{0, 1 \ldots v_j\}$	ADV for jth number feature ($j \in [1, m]$)

Feature derivation: The feature derivation block receives the raw data from the previous block and extracts the required features. The derivation of some features might demand further calculation, like the packet interval and the keystroke press interval. After each feature is ready for processing, the feature derivation block passes the data to the next block.

Hashing-encryption: The hashing-encryption block is responsible for generating a cryptographic profile based on all of the available features via fuzzy hashing and FHE. 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 $\dot{P} \leftarrow \{\dot{P}_F, \dot{P}_H\}$. In addition, the feature derivation block continues to produce feature information at the end of every window period; therefore, the hashing-encryption block generates a new cryptographic profile based on the latest incoming feature information and overwrites the former one.

Furthermore, as illustrated in Table 3, the features are divided into two categories: host-based and network-based, which are further classified into several subcategories as follows.

File system and registry: This subcategory consists of fuzzy hashes of the file hierarchy at a critical path and a portion of the registry contents (for Windows systems). In particular, we target the information pertaining to the installed software since, for most users, the installed software is relatively stable and is a good representation of user habits compared to other attributes.

For the Windows OS, the folder path for installed software is typically "C:\Program Files" and "D:\Program Files" while for the Linux OS, it is "/usr/bin/." Besides, the Windows Registry is a hierarchical database which stores configuration settings and options for both low-level OS components and high-level running applications, which reflects users' utilization of software well. In our case, we only retrieve the part of the registry which contains the information of the installed software

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. According to Reference 8, a reasonable choice of the window size is 20 mouse clicks, which takes up to approximately 3.03 minutes

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. Kevin Killourhy and Roy Maxion described in Reference 38 a method by which timing data for keystroke activity of typing a password is collected and used to classify impostors from legitimate users. Their experimental results show that these two-time metrics are sufficient to represent different users. In our proposed system, the timing metrics of keystroke activity are captured and derived, while a user is typing his password in order to log into the operating system

System processes: This subcategory is composed of fuzzy hashes of four clusters of system process names. The clustering strategy is alphabetically and then divides the names into four blocks

Browser information: In this subcategory, we utilize the autofill information in browsers. We derive the fuzzy hash of personal information with attributes of "Email," "Username," and "Address." The significance of autofill information is that one tends to have the same autofill value for those frequently used attributes, in different browsers or hosts

Flow-based features: We model users' general network behavior via 19 flow-based features. See Appendix B for the detailed definition of the features used to profile a user's network behavior. Note that it is commonly known that flow-based features indicate the category of network traffic (eg, 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

4.2.2 | Algorithm for user profile generation

From Table 3, we observe that the user characteristics can be classified as two types: string (Sections 1, 4, 5) and numerical (Sections 2, 3, 6) values. Due to the respective working mechanisms of fuzzy hashing and homomorphic encryption, we apply the former on string characteristics and the latter on numerical ones. Suppose each user has n string characteristics as well as m number characteristics where each string feature is denoted by $p_{F_i}(i \in [1, n])$ and each numerical feature is denoted by $p_{H_j}(j \in [1, m])$. Also suppose that the user has already generated one pair of homomorphic encryption keys denoted by (pk, sk). The process of generating a cryptographic user profile is given in Algorithm 1. The notations used in Algorithm 1 are presented in Table 4. 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.

4.3 | Enrollment of first-time user and profile update

The sequence diagram of a first-time user's *enrollment* is shown in Figure 4. To initiate the *enrollment*, a user shares his public key pk with the AS (message 1). After receiving the server's acknowledgement (message 2), the user passes his uid, Psw along with his cryptographic user profile $\dot{P}$. Now, the server has the user's cryptographic profile stored in the database. As mentioned earlier, the user profiles change constantly. Thus, the cumulative change of user profiles makes it harder to recognize the distance between the newly captured profile and the original one stored in the database. To address this issue, the UPDB should update the cryptographic user profile for each user, once every time period T. When updating a user's profile, the AS first authenticates the user as discussed in Section 4.4. After the user is validated, the profile update proceeds similar to the process of enrollment, which is shown in Figure 4. The system administrator is responsible for setting a reasonable T when deploying PINTA.

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

4.4 | Server authentication

The sequence diagram of the server handling each authentication request is presented in Figure 5. To initiate an authentication attempt, the user passes the uid, Psw' for the first-factor authentication (message 1). The failure of first-factor authentication

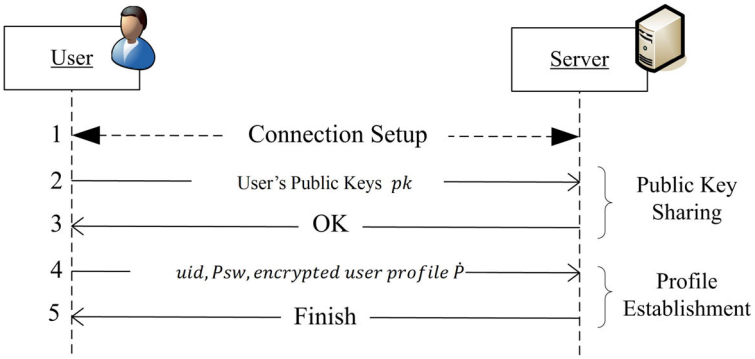

FIGURE 4 Sequence diagram for user enrollment

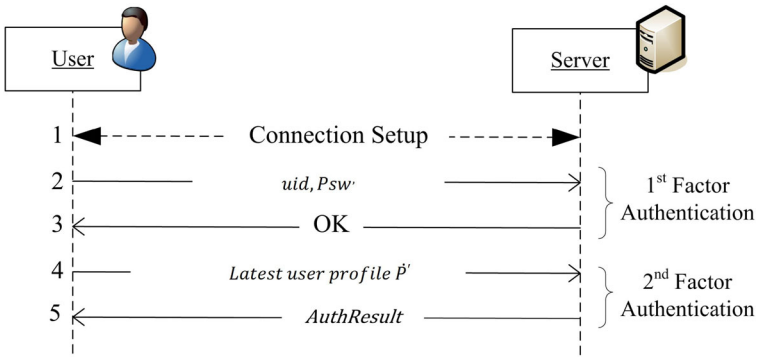

FIGURE 5 Sequence diagram for authentication

terminates the conversation. If the user passes the first step of the authentication, he then passes his newly generated user profile, denoted by $\dot{P}'$ (message 3), to the server. After evaluating the distance between $\dot{P}'$ and $\dot{P}$, the server returns a boolean value *AuthResult* (message 4), 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).

ADV is used as the array of all ADVs, which is assigned by the server for every numerical characteristic. In our system, for the *j*th numerical feature, we propose a threshold k_j of the distance value. We define ADV V_j as the array of all integers ranging from 0 to k_j (ie, $V_j \leftarrow \{0, 1 \dots k_j\}$). In addition, we place different weights on each string feature and each numerical feature, denoted by W_F and W_H, respectively. We will address the problem of how to reasonably set the threshold and weight for each feature in Section 5.3. The process of estimating profile distance and calculating *AuthResult* is given in Algorithm 2 and the notations used are presented in Table 3.

Algorithm 1 Generate Cryptographic User Profile()

Require: P_F, P_H, pk
Ensure: $\dot{P} \leftarrow \{\dot{P_F}, \dot{P_H}\}$
  for each $i \in [1, n]$ **do**
    $\dot{p_{F_i}} \leftarrow FuzzyHash(p_{F_i})$;
  end for
  for each $j \in [1, m]$ **do**
    $\dot{p_{H_j}} \leftarrow FHE\_Encrypt(p_{H_j}, pk)$;
  end for
  $\dot{P_F} \leftarrow \{\dot{p_{F_1}} \dots \dot{p_{F_i}} \dots \dot{p_{F_n}}\}$;
  $\dot{P_H} \leftarrow \{\dot{p_{H_1}} \dots \dot{p_{F_j}} \dots \dot{p_{F_m}}\}$;
  $\dot{P} \leftarrow \{\dot{P_F}, \dot{P_H}\}$;
  return $\dot{P}$;

TABLE 5 Data source for experiments

Section	Dataset	Subjects[a]
Mouse dynamic	NSKEYLAB Dataset[13]	10
Keystroke activity	CMU Dataset[14]	51
General network	DACS Dataset[12]	132

[a]Number of subjects involved in the dataset.

Algorithm 2 Calculate Authentication Result()

Require: $\dot{P}, \dot{P}', n, m, pk, W_F, W_H, V_j$
Ensure: *AuthResult*: true—false
initial $Dis \leftarrow 0$
for each $i \in [1, n]$ **do**
$dis_{F_i} \leftarrow FuzzyCmp(\dot{p_{F_i}}, \dot{p_{F_i}}')$;
$Dis \leftarrow Dis + dis_{F_i} * w_{F_i}$;
end for
for each $j \in [1, m]$ **do**
$dis_{H_j} \leftarrow$
$FHE\_Div(FHE\_SUB(\dot{p_{H_j}}', \dot{p_{H_j}}, pk), \dot{p_{H_j}}, pk)$;
for each $v \in V_j$ **do**
if $dis_{H_j} == FHE\_Encrypted(v, pk)$ **then**
$dis_{H_j} = 1$; break;
end if
$dis_{H_j} = -1$;
end for
$Dis \leftarrow Dis + dis_{H_j} * w_{H_j}$;
end for
$AuthResult \leftarrow (Dis > 0)\ ?\ true; false$;
return *AuthResult*;

5 | EXPERIMENTATION AND EVALUATION

In this section, we evaluate the feasibility of our proposed system through a series of experiments. We specifically conducted experiments with a combination of several public datasets and a dataset which we generated. Particularly, we evaluate PINTA in terms of authentication performance, the overhead caused on the system, and computational performance.

5.1 | Datasets and user profile generation

The names of the public datasets used in our experiments are shown in Table 5.

A description of the datasets, along with a description of the necessary preprocessing approach for each dataset is as follows. (a) *NSKEYLAB Dataset*[13] 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.[13,39] We derived three angle-based metrics using the approach proposed in Reference 8, thus, generating three data points that represent the user's mouse movement profile. (b) *CMU Dataset*[14] is a dataset consisting of keystroke-timing information from 51 subjects (typists), each typing a password 400 times.[14,38] (c) *DACS Dataset*[12] is a dataset consisting of the network trace for an educational organization. A 100 Mbit/s Ethernet link connecting the organization to the Internet was monitored to generate the trace. Each user at this site is assigned a fixed IP address.[12] We used SplitPcap[40] to obtain separate pcap files for each IP address inside the

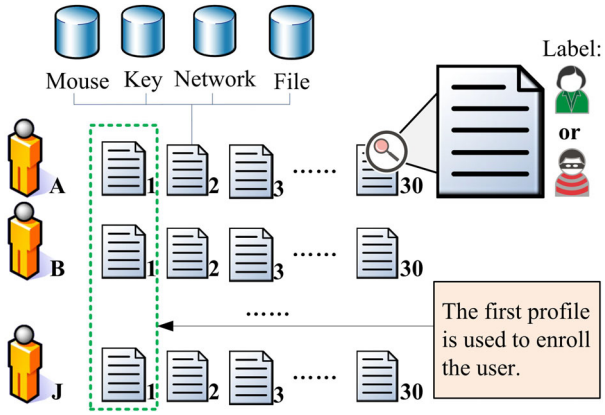

FIGURE 6 The generation of hybrid user profile samples

organization's network and further partition each pcap file into smaller pcap files with a period of 30 minutes. For each small pcap file, we ran TSTAT[37] to derive the flow-based features, thus, generating the user's network behavior profile. This dataset spans 2 months.

In addition to these three datasets, we generated the dataset that contains *file system* and *registry* information as follows. First, we generated two *Software Pools*, A and B, which are two lists of software names with 30 and 100 types of software. Software Pool A represents the OS preinstalled software and common software, while Pool B represents personalized software. For 15 users, we randomly chose 20 software names from Pool A and five from Pool B for each user. We fuzzy hashed the list of the software names, thereby obtaining the first piece of the profile for each individual. Then, for each user, we randomly chose two uninstalled software packages from Pool B and inserted them into the user's original software list, thus generating a new piece of the user profile. We conducted this procedure iteratively for 30 times in order to generate a dataset representing the "file system" with 15 individuals, each of which, has 30 profiles. We generated two software pools for two reasons: First, in the initial stage, users have similar software environments, in which most installed software is preinstalled with the OS by default and some are commonly used software. Second, the software environment for users tends to diverge afterward according to users' habits, interests, and occupations. In our approach, *Software Pool* A represents OS preinstalled and common software while *Software Pool* B represents personalized software.

Based on these four datasets, we generated hybrid user profiles for each user. As shown in Table 5, the NSKEYLAB Dataset only contains data from 10 participants. Therefore, we can at most generate user profiles for 10 distinctive subjects. To generate one piece of the hybrid user profile for a certain user, we first randomly chose four profiles, each from one dataset. Then, we combined 30-minute chunks from each of the four datasets to create 30 hybrid profiles belonging to one user. Hence, in our experiments, thirty 30-minute hybrid profile samples are created per user. The users are labeled as User A, User B, and so on. Finally, we eliminated the previously used profiles from the four datasets. By repeating this procedure, we generated 30 hybrid user profile samples for each of the 10 users, yielding 300 different hybrid profile samples, as illustrated in Figure 6.

5.2 | Experimental setup

For each user, we registered his or her first piece of the 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 mislabeled them making the actual owner of that profile appears 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 encrypted each profile in the testing set via Algorithm 1 and left the profiles in the training set as plaintext (the data from this training set can result from a bank's focus group). The training set serves as a priori knowledge for threshold setting and weight adjustment while the testing set was the input of the system during experiments for performance evaluation. We first performed a threshold setting and weight adjustment based on the training set, as explained in Section 5.3. Then, we conducted a series of experiments by iteratively comparing the result generated by Algorithm 2 with the actual situation for each piece of the user profiles. We conducted all the experiments on a standard laptop computer with Intel CPU i5-M430 (2.27GHz) and RAM of 4 GB.

5.3 | Decision process

To decide whether a hybrid user profile is legitimate, we must first determine 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 in this section.

5.3.1 | Threshold setting

We used the following approach to find an optimal threshold for each feature based on the a priori knowledge we had. Assume there are p features in each hybrid user profile and altogether q hybrid profile samples as prior knowledge. d_{ij} denotes the change percentage of the jth feature in the ith profile compared to the enrollment profile stored in the database. l_i denotes the identity of the owner of the ith profile. $l_i = 1$ if the owner is legitimate and $l_j = 0$ if the owner is an intruder. th_j denotes the threshold for the jth feature. Thus, for the jth feature, we design a function F of th_j:

$$\begin{aligned} F(th_j) &= l_1 * Sgn(th_j - d_{1j}) + \ \ldots \ l_i * Sgn(th_j - d_{ij}) \\ &\quad + . \ \ldots \ l_n * Sgn(th_j - d_{nj}) \\ &= \sum_{i=1}^{q} l_i * Sgn(th_j - d_{ij}), \end{aligned} \tag{5}$$

where $F(th_j)$ represents the total number of correctly classified cases for the jth feature and Sgn denotes the sign operation. Thus, the optimal th_j is the one that makes $F(th_j)$ reach a relatively maximum value. In this way, the threshold setting problem is transformed to a simple linear function maxima problem, which can be easily solved by Matlab.

5.3.2 | Weight adjustment

After having a vector of optimal thresholds for all the features, each feature is used as a voter, either voting for or against the user. The weight placed on each vote is adjusted via linear programming as described in Equation (6), where $v_j \in \{1, 0\}$ denotes the vote of the jth feature and $rst \in \{1, 0\}$ denotes the actual result.

$$\min \sum_{j=1}^{p} w_j * v_j - rst, s.t. \sum_{j=1}^{p} w_j = 1. \tag{6}$$

After the implementation of linear programming with Matlab, we identified the features that were assigned far more weight than others, which indicates that those features better represent user behavior. See Appendix C for the weight adjustment results. Note that in the deployment of our system, the system administrator can always modify the weight adjustment based on his knowledge/experience. Additionally, he can eliminate the feature with little or zero weight and add new features to the system, which makes PINTA highly configurable and robust.

5.4 | Results

In this section, we evaluate PINTA in terms of authentication performance, system overhead, computational performance, and present our results.

5.4.1 | Authentication performance

In order to evaluate the PINTA's authentication performance, we used recall, FPR as a metric. We define recall and FPR as follows:

- *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*. Thus, recall indicates the authentication system's ability to identify intruders.

TABLE 6 Experiment results: Recall and FPR

Time for data collection	Weight adjustment	Recall (%)	FPR (%)
5 minutes	Without	74.2	26.9
	With	75.1	24.3
10 minutes	Without	75.4	19.7
	With	77.1	19.3
20 minutes	Without	78.8	16.2
	With	79.9	15.1
30 minutes	Without	80.8	14.7
	With	82.7	13.2

- *False positive rate*: 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. FPR refers to the probability of the system's falsely rejecting a legitimate user.

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 and with or without weight adjustment. The experimental results in terms of recall and FPR are presented in Table 6. 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 82.7% and FPR of 13.2%. 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. Also, it can be observed that weight adjustment makes a positive impact on the system performance via a relatively higher recall and a relatively lower FPR.

5.4.2 | System overhead and utilization

In order to evaluate the overhead caused by PINTA on the system and the utilization of PINTA, we computed packet size, system overhead, and resource utilization such as CPU and RAM utilization. We define and calculate them as follows:

- 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.
- *System overhead*: System overhead is the initial latency during the entire MFA process. Measuring total system overhead is a complex task. In Section 5.4, we evaluated the overhead introduced by the user profile acquisition, cryptography, data transmission, and server processing.
- *Resource utilization*: Resource utilization is defined as how much system resources it takes for a user information acquisition program to retrieve information and derive features continuously. We evaluated it in terms of CPU and RAM 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 (7), 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.

$$Size = pkSize + n * \alpha + m * \beta. \tag{7}$$

In PINTA, with $pkSize = 128\,KB$, $n = 9$, $\alpha = 0.125\,KB$, $m = 24$, and $\beta = 128\,KB$, the packet size for the enrollment is approximately 3201 KB (message 1 and message 3 in Figure 4). In the authentication procedure, the user is not expected to transmit their pk again; so, the packet size is around 3201 KB after enrollment (message 3 in Figure 5).

System overhead, T, is comprised of the time spent on authentication data transmission, T_t and the server process, T_s. We can derive T using Equation (8). 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

TABLE 7 Average timing results

GenerateCryptographicUserProfile				*CalculateAuthenticationResult*			
FHE_KeyGen	***FuzzyHash***	***FHE_Encrcypt***		***FuzzyCmp***	***FHE_Sub***	***FHE_Div***	***FHE_Decrypt***
1449 microseconds	677 microseconds	47 784 microseconds		57 microseconds	1392 microseconds	370 728 microseconds	29 040 microseconds
			Subtotal = 0.05 seconds				Subtotal = 0.37 seconds

request is around 35 seconds. It is worth mentioning that system overhead may vary because it is dependent on the computing ability of both the client and the server sides and also to the specific network conditions.

$$T_{\text{after}} = T_t + T_s. \tag{8}$$

In terms of resource utilization, in our experiments, the running of the user PAP takes less than 3% of CPU resources and about 15 MB of RAM. The most significant 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. Note that the statistic 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 machine.

5.4.3 | Computational performance

Finally, we measure PINTA's computational performance. For this purpose, we used the timing results of two main algorithms: *GenerateCryptographicUserProfile*() and *CalculateAuthenticationResult*(). Moreover, we also computed the timing results of operations in Table 2.

CalculateAuthenticationResult includes subtraction and division operations. While performing the subtraction, while we used the built-in subtraction function in the homomorphic library, the division is implemented by multiplying the reverse of the second multiplier. Therefore, each division is indeed the combination of encryption and multiplication.

The results are given in Table 7. According to the results, while the generation of cryptographic profiles takes around 0.05 seconds, the calculation of authentication result takes about 0.37 seconds. When we look at the operations performed in these algorithms, we see that the division operation dominates the authentication result calculation algorithm. That's because the division operation is a multiplication operation and the homomorphic multiplication operations are computationally much more expensive than the addition. However, still, the results show that PINTA is feasible in terms of computational feasibility.

5.5 | Security analysis

In this subsection, we demonstrate how the authentication system thwarts the adversaries discussed in Section 3.1.

Security against brute-force attacker: In this case, an adversary attempts to log in by guessing a user's profile, which, in the worst case, would involve traversing the entire message space. For each fuzzy hash, the message space is 2^{128}. For each FHE, the message space is $2^{1.5\cdot 10^5}$. Since the message space for the fuzzy hash is arbitrarily small compared to that of FHE ciphertext, we only consider the computational cost to brute force the FHE ciphertext. First, denoting the length of ciphertext by l, the probability of a malicious code correctly forging one correct feature is:

$$P_{\text{forge}} = \frac{1}{2^l}. \tag{9}$$

In our system, we realize $l = 1.5 \ldots 10^5$, so P_{forge} is $\frac{1}{2^{1.5\cdot 10^5}}$. Second, an adversary that is brute forcing the FHE ciphertext for each feature will have to put an overall effort of

$$\Psi = \frac{\gamma}{\kappa}, \tag{10}$$

where γ is the average number of possible ciphertext values (eg, $2^{1.5\cdot 10^5}$) and κ is the frequency of the attacker's computer at 1 *attack*/*s*. Assuming an attacker with computational resources such as Intel Core i7-3960 at 3.9GHz, approximately $2^{1.499\cdot 10^5}$ years of Ψ would be needed to generate the correct ciphertext. Therefore, it is safe to conclude that it is impossible for a computationally bounded adversary to match a user's profile by brute force.

Security against honest-but-curious server attacker: In this scenario, the *honest-but-curious server* tries to reverse engineer a user's cryptographic profile $\dot{P}'$. The capability of resisting reverse engineering is also referred to as *semantic security* in

cryptology.[41] Informally, a traditional definition of *semantic security* is that a system is semantically secure if any computationally bounded adversary is not able to compute the plaintext even with the knowledge of both ciphertext and the corresponding public key. Note that in our work, we also treat fuzzy hashing as a special form of encryption. Because each $\dot{P}'$ comprises a piece of fuzzy-hashed user profile $P_H{}'$ and a piece of homomorphically encrypted user profile $P_F{}'$, we demonstrate the semantic security of fuzzy hashing and FHE, respectively. For fuzzy hashing, the only available information available to the *honest-but-curious*[42] server or any third-party adversary are the two fuzzy hashed values (after the initial authentication attempt) and the only answer the server can obtain is the similarity between two fuzzy hashes, which is just as intended. No further information can be derived from the hash values due to the one-way property of fuzzy hashing.[10] For FHE, Gentry and Coron showed that an FHE system is semantically secure,[11,29] which means it is infeasible for a computationally bounded adversary to derive significant information about a message (plaintext) when given only its ciphertext and the corresponding public key.

6 | CONCLUSION

In this work, we propose a privacy-preserving MFA system, called PINTA. In PINTA, while the first authentication factor is a password and the second one is a hybrid behavioral user profile. PINTA focuses on the privacy preservation of the second factor, which has two advantages over previously proposed systems. First, user privacy is not leaked to the AS. We have proven that the system is secure against both a brute-force matching attacker and an honest-but-curious attacker. Second, the hybrid user profiling model is highly usable and configurable. One can always modify the feature list for user profiling in PINTA according to the actual circumstances. We evaluated the system performance via a series of experiments, resulting in an optimal recall of 82.7% and FPR of 13.2%. In addition, we also show that PINTA's system overhead is within the acceptable range. We plan to extend PINTA to allow a user to have one profile that is valid for multiple machines. One way of achieving this is to place less weight on features that are likely unique to a machine (eg, network dynamics) and more weight on features that are likely unique to users (eg, browser data).

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers. This work is partially supported by US National Science Foundation (NSF) under the grant numbers NSF-CNS-1718116 and NSF-CAREER-CNS-1453647.

CONFLICT OF INTEREST

The authors declare that there are no conflicts of interest.

ORCID


Abbas Acar https://orcid.org/0000-0002-4891-160X

REFERENCES

1. AWE Multi-Factor Authentication Frequently-Asked Questions 2012. http://aws.amazon.com/cn/mfa/faqs/. Accessed June 01 2018.
2. About two-step Verification. 2012. http://support.google.com/accounts/bin/topic.py?topic=28786. Accessed June 01 2018.
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/. Accessed June 01 2018.
4. Paul C, Morse E, Zhang A, Choong YY, Theofanos M. *A field study of user behavior and perceptions in smartcard authentication*. *6949 of Lecture Notes in Computer Science*. IFIP Conference on Human-Computer Interaction - INTERACT 2011. Springer. 2011 (pp. 1–17).
5. Validation and ID Protection. 2012. https://idprotect.verisign.com/mainmenu.v. Accessed June 01 2018.
6. Sujithra M, Padmavathi G. Next generation biometric security system: an approach for mobile device security. Paper presented at: Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology; 2012: 377–381.
7. Jorgensen Z, Yu T. On mouse dynamics as a behavioral biometric for authentication. Paper presented at: *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*; 2011: 476–482.
8. Zheng N, Paloski A, Wang H. An efficient user verification system via mouse movements. Paper presented at: Proceedings of the 18th ACM conference on Computer and communications security; 2011: 139–150.
9. Strasburg C, Krishnan S, Dorman K, Basu S, Wong JS. Masquerade detection in network environments. Paper presented at: 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet; 2010: 38–44. https://doi.org/10.1109/SAINT.2010.66
10. Kornblum J. Identifying almost identical files using context triggered piecewise hashing. Paper presented at: The Proceedings of the 6th Annual Digital Forensic Research Workshop (DFRWS '06). *Digital Investigation*; 2006; 3, Supplement(0): 91–97.
11. Gentry C. *A fully homomorphic encryption scheme*. [PhD thesis]. Stanford University; 2009.

12. Pcap Trace: Trace 6, *Packet Headers*. 2012. http://traces.simpleweb.org. Accessed June 01 2018.
13. Mouse Data for Continuous Authentication. 2012. http://nskeylab.xjtu.edu.cn/people/cshen/. Accessed June 01 2018.
14. Keystroke Dynamics: *Benchmark Data Set*. 2012. http://www.cs.cmu.edu/~keystroke. Accessed June 01 2018.
15. Bonneau J, Herley C, Oorschot PCv, Stajano F. The quest to replace passwords: a framework for comparative evaluation of web authentication schemes. Paper presented at: *Proceedings of the 2012 IEEE Symposium on Security and Privacy*; 2012: 553–567.
16. RSA SecurID - World's Leading Two-Factor Authentication. 2012. http://www.emc.com/security/rsa-securid.htm. Accessed June 01 2018.
17. Kumar M. New remote user authentication scheme using smart cards. *IEEE Trans Consum Electron*. 2004;50(2):597-600.
18. Czeskis A, Dietz M, Kohno T, Wallach D, Balfanz D. Strengthening user authentication through opportunistic cryptographic identity assertions. Paper presented at: CCS '12 Proceedings of the 2012 ACM conference on Computer and communications security; 2012: 404–414.
19. Yampolskiy RV, Govindaraju V. Behavioural biometrics: a survey and classification. *Int J Biometrics*. 2008;1(1):81-113.
20. Bhargav-Spantzel A, Squicciarini A, Bertino E. Privacy preserving multi-factor authentication with biometrics. Paper presented at: Proceedings of the Second ACM Workshop on Digital Identity Management; 2006: 63–72.
21. Salem A, Obaidat MS. A novel security scheme for behavioral authentication systems based on keystroke dynamics. *Secur Privacy*. 2019;2(2):e64. https://doi.org/10.1002/spy2.64.
22. Tasia CJ, Chang TY, Cheng PC, Lin JH. Two novel biometric features in keystroke dynamics authentication systems for touch screen devices. *Secur Commun Network*. 2014;7(4):750-758. https://doi.org/10.1002/sec.776.
23. Kambourakis G, Damopoulos D, Papamartzivanos D, Pavlidakis E. Introducing touchstroke: keystroke-based authentication system for smartphones. *Secur Commun Network*. 2016;9(6):542-554. https://doi.org/10.1002/sec.1061.
24. Liu W, Uluagac AS, Beyah R. MACA: a privacy-preserving multi-factor cloud authentication system utilizing big data. Paper presented at: 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); 2014: 518–523. https://doi.org/10.1109/INFCOMW.2014.6849285
25. IETF. Transport Layer Security (tls). 2008. http://datatracker.ietf.org/wg/tls/. Accessed June 01 2018.
26. ssdeep. 2012. http://ssdeep.sourceforge.net/. Accessed June 01 2018.
27. Bayer U, Comparetti PM, Hlauschek C, Kruegel C, Kirda E. *Scalable, Behavior-Based Malware Clustering*. In The Network and Distributed System Security Symposium (NDSS). Vol 9; 2009:8-11.
28. Dijk VM, Gentry C, Halevi S, Vaikuntanathan V. Fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2009/616; 2009. http://eprint.iacr.org/.
29. Coron JS, Mandal A, Naccache D, Tibouchi M. *Fully homomorphic encryption over the integers with shorter public keys*. In: Rogaway P, ed. *Advances in Cryptology, Annual Cryptology ConferenceCRYPTO 2011. 6841 of Lecture Notes in Computer Science*. Springer; 2011:487-504.
30. Brakerski Z, Vaikuntanathan V. Fully homomorphic encryption from ring-LWE and security for key dependent messages. Annual Cryptology Conference; 2011: 505–524.
31. Brakerski Z. *Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP*. Springer; Annual Cryptology Conference; 2012:868-886.
32. Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans Comput Theory*. 2014;6(3):13.
33. Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J Comput*. 2014;43(2):831-871.
34. Acar A, Aksu H, Uluagac AS, Conti M. A survey on homomorphic encryption schemes: theory and implementation. *ACM Comput Surv*. 2018;51(4):79:1-79:35. https://doi.org/10.1145/3214303.
35. Simple Encrypted Arithmetic Library (release 3.1.0). 2018. Microsoft Research, Redmond, WA. https://github.com/Microsoft/SEAL. Accessed June 01 2018.
36. Fan J, Vercauteren F. Somewhat practical fully homomorphic encryption. Paper presented at: IACR Cryptology ePrint Archive 2012; 2012: 144.
37. TSTAT- Tcp STatistic and Analysis Tool. 2012. http://tstat.tlc.polito.it/index.shtml. Accessed June 01 2018.
38. Killourhy K, Maxion R. Comparing anomaly-detection algorithms for keystroke dynamics. Dependable Systems Networks, 2009. DSN '09. Paper presented at: IEEE/IFIP International Conference on 2009: 125–134.
39. Shen C, Cai Z, Guan X. Continuous authentication for mouse dynamics: A pattern-growth approach. Dependable Systems and Networks (DSN), Paper presented at: 2012 42nd Annual IEEE/IFIP International Conference on 2012: 1–12.
40. NETRESEC SplitCap - A fast PCAP file splitter. 2012. http://www.netresec.com/?page=SplitCap. Accessed June 01 2018.
41. Goldwasser S, Micali S. Probabilistic encryption. *J Comput Syst Sci*. 1984;28(2):270-299.
42. Abbasinezhad-Mood D, Nikooghadam M. Design and extensive hardware performance analysis of an efficient pairwise key generation scheme for smart grid. *Int J Commun Syst*. 2018;31(5):e3507. e3507 IJCS-17-0460.R2. https://doi.org/10.1002/dac.3507.

AUTHOR BIOGRAPHIES

Abbas Acar is currently a Graduate Research Assistant in the Department of Electrical & Computer Engineering, Florida International University (FIU), USA and pursuing his PhD degree in the same department. He received the BS degree in Computer Engineering from Middle East Technical University (METU), Ankara, Turkey in 2015.

Wenyi Liu is currently a master candidate from both The Georgia Institute of Technology and Shanghai Jiao Tong University. Her major is Electrical and Computer Engineering. She received a BS in Information Security from Shanghai Jiao Tong University. The focus of her research lies at network security.

Raheem Beyah is an Associate Professor in the School of Electrical and Computer Engineering at Georgia Tech where he leads the Georgia Tech Communications Assurance and Performance Group and is a member of the Georgia Tech Communications Systems Center (CSC). Prior to returning to Georgia Tech, Dr Beyah was an Assistant Professor in the Department of Computer Science at Georgia State University, a research faculty member with the Georgia Tech CSC, and a consultant in Andersen Consulting's (now Accenture) Network Solutions Group. He received his Bachelor of Science in Electrical Engineering from North Carolina A&T State University in 1998. He received his Masters and PhD in Electrical and Computer Engineering from Georgia Tech in 1999 and 2003, respectively. Dr Beyah served as a Guest Editor for MONET. He is an Associate Editor of several journals including the (Wiley) Wireless Communications and Mobile Computing Journal. His research interests include network security, wireless networks, network traffic characterization and performance, and security visualization. He received the National Science Foundation CAREER award in 2009 and was selected for DARPA's Computer Science Study Panel in 2010. He is a member of NSBE, ASEE, and a senior member of ACM and IEEE.

Kemal Akkaya is an Associate Professor in the Department of Electrical and Computer Engineering at Florida International University. He received his PhD in Computer Science from University of Maryland Baltimore County in 2005 and joined the Department of Computer Science at Southern Illinois University (SIU) as an Assistant Professor. Dr Akkaya was an Associate Professor at SIU from 2011 to 2014. He was also a visiting professor at The George Washington University in Fall 2013. His current research interests include security and privacy, energy-aware routing, topology control, and quality of service issues in a variety of wireless networks such as sensor networks, multimedia sensor networks, smartgrid communication networks, and vehicular networks. Dr Akkaya is a senior member of IEEE. He is the area editor of Elsevier Ad Hoc Network Journal and serves on the editorial board of IEEE Communication Surveys and Tutorials. He has served as the guest editor for Journal of High-Speed Networks, Computer Communications Journal, Elsevier Ad Hoc Networks Journal and in the TPC of many leading wireless networking conferences including IEEE ICC, Globecom, LCN, and WCNC. He has published over 90 papers in peer-reviewed journal and conferences. He has received "Top Cited" article award from Elsevier in 2010.

A. Selcuk Uluagac is currently an Assistant Professor in the Department of Electrical and Computer Engineering (ECE) at Florida International University (FIU). Before joining FIU, he was a Senior Research Engineer in the School of Electrical and Computer Engineering (ECE) at Georgia Institute of Technology. Prior to Georgia Tech, he was a Senior Research Engineer at Symantec. He earned his PhD with a concentration in information security and networking from the School of ECE, Georgia Tech in 2010. He also received an MSc in Information Security from the School of Computer Science, Georgia Tech and an MSc in ECE from Carnegie Mellon University in 2009 and 2002, respectively. The focus of his research is on cyber security topics with an emphasis on its practical and applied aspects. He is interested in and currently working on problems pertinent to the security of Cyber-Physical Systems and Internet of Things. In 2015, he received a Faculty Early Career Development (CAREER) Award from the US National Science Foundation (NSF). In 2015, he was awarded the US Air Force Office of Sponsored Research (AFOSR)'s 2015 Summer Faculty Fellowship. He is also an active member of IEEE (senior grade), ACM, and ASEE and a regular contributor to national panels and leading journals and conferences in the field. Currently, he is the area editor of Elsevier Journal of Network and Computer Applications and serves on the editorial board of the IEEE Communication Surveys and Tutorials. More information can be obtained from: http://web.eng.fiu.edu/selcuk.

How to cite this article: Acar A, Liu W, Beyah R, Akkaya K, Uluagac AS. A privacy-preserving multifactor authentication system. *Security and Privacy*. 2019;2:e88. https://doi.org/10.1002/spy2.88

APPENDIX A: FUZZY HASHING EXPERIMENT

We conducted a simple experiment to show the efficacy of fuzzy hashing and how fuzzy hashed strings can be compared to gauge their level of similarity. The procedure of the experiment is as follows: (a) We used a newly installed (clean) Windows 7 OS and 10 different software installation packages, labeled as A, B, … J. (b) We installed Software A on the clean OS and used the **tree** command to output the file hierarchy in the folder path "C:Program Files" as a string, denoted by Str_1. (c) We installed 10 software packages one by one and obtained 10 strings, denoted Str_1, Str_2 … Str_10. Then, we generated the fuzzy hash for each string using *ssdeep*, denoted by SD_1, SD_2 … SD_10. We computed the *similarity score* (SD_{Score}) between every hash generated by *ssdeep*. Since we had 10 fuzzy hashes, there are 100 combinations in total. The SD_{Score} for each combination is presented in Table A1. From Table A1, we make several observations. First, all the similarity scores are diagonally symmetric because SD_{Score} (SD_1, SD_2) and SD_{Score}(SD_2, SD_1) are the same. Second, the similarity score for the fuzzy hashes of the same string is always 100. Third, with the installation of more software (ie, more of a change to the file hierarchy and corresponding string), the similarity score reduces. Therefore, we can say that the *similarity score* of fuzzy hashes can roughly represent how similar two fuzzy hashed strings are.

APPENDIX B: FEATURES USED TO PROFILE USER NETWORK BEHAVIOR

We profiled users' network behavior via the network flow-based features listed in Table B1. Note that except "flowDuration," each feature is measured on both client-to-server direction and server-to-client direction. Also, the value of each feature is based on one flow. A flow is defined as a sequence of packets sent from a particular source to a particular destination. Taking feature "stddevRTT" as an example, it represents the standard deviation of all the round-trip times in that flow.

Output options

APPENDIX C: WEIGHT ADJUSTMENT RESULT

The result for weight adjustment in Section 5.3.2 is as follows: (a) For the mouse movement subcategory, the weights for *direction*, *curvature distance*, and *curvature angle* were 0.106, 0.035, and 0.004, respectively. (b) For the keystroke activity subcategory, the weights for *key press-down time* and *average key-press interval* were 0.048 and 0.023. (c) For the general network section, the weight for each feature is shown in Table B1. (d) For the file system and registry, the weights for the *file system* feature and *registry* feature are 0.132 and 0.129. We have 26 features in total and the sum of the weights equals to 1.

TABLE A1 Similarity score between fuzzy hashes. The element in the *i*th row and *j*th column represents the value of $SD_{Score}(SD_i, SD_j)$

	1	2	3	4	5	6	7	8	9	10
1	100	89	89	74	63	62	62	40	40	40
2	89	100	86	76	76	55	55	45	45	45
3	89	86	100	92	82	77	58	58	48	38
4	74	76	92	100	90	90	70	60	50	50
5	63	76	82	90	100	81	81	72	62	62
6	62	55	77	90	81	100	91	81	72	72
7	40	55	58	70	81	91	100	91	81	81
8	40	45	58	60	71	81	91	100	92	92
9	40	45	48	50	62	72	81	92	100	100
10	40	45	38	50	62	72	81	92	100	100

TABLE B1 Network flow-based features

No	Identifier	Definition	Weight
1,2	noPacket	Number of packets	0.012, 0.007
3,4	maxSegment	Maximum TCP segment	0, 0.001
5,6	maxWindow	Maximum TCP Window	0.001, 0
7,8	minWindow	Minimum TCP window	0, 0
9,10	avgRTT	Average round-trip time	0.045, 0.032
11,12	stddevRTT	Stddev of round-trip time	0.003, 0
13,14	minTTL	Minimum time-to-live	0, 0.005
15,16	maxTTL	Maximum time-to-live	0, 0.005
17,18	avgInterPkt	Average packet arrival	0.134, 0.119
19	flowDuration	Flow completion time	0.087