Privacy-Preserving Implicit Authentication

Nashad Ahmed Safa[1], Reihaneh Safavi-Naini[1], and Siamak F. Shahandashti[2]

[1] University of Calgary, Canada
{rei,nasafa}@ucalgary.ca
[2] Newcastle University, UK
siamak.shahandashti@ncl.ac.uk

Abstract. In an implicit authentication system, a user profile is used as an additional factor to strengthen the authentication of mobile users. The profile consists of features that are constructed using the history of user actions on her mobile device over time. The profile is stored on a server and is used to authenticate an access request originated from the device at a later time. An access request will include a vector of recent features measurements on the device that will be matched against the stored features to accept or reject the request. The features however include private information such as user location or web sites they have visited. In this paper we propose *privacy-preserving implicit authentication* which achieves implicit authentication without revealing unnecessary information about the users' usage profiles to the server. We propose an architecture, give formal security models, and propose constructions with provable security. We consider two security models, namely for cases where the device behaves semi-honestly or maliciously.

Keywords: Implicit Authentication, User Privacy, Homomorphic Encryption, Provable Security, Behavioural Features.

1 Introduction

In mobile applications such as mobile commerce, users often provide authentication information using Mobile Internet Devices (MIDs) including cell phones and notebooks. In most cases however, password authentication is the primary method of authentication. The weaknesses of password-based authentication systems, including widespread usage of weak passwords, have been widely studied (see e.g. [25] and references within). In addition to these weaknesses, limitations of user interface on MIDs results in an error prone process for inputting passwords, encouraging even poorer choices of password by users.

To strengthen authentication, two-factor authentication has been proposed. The second factor, when based on extra hardware such as SecureID tokens, have additional cost and limit their wide application. An attractive method of strengthening password systems is *implicit authentication* [13] which effectively adds a second factor to authentication. The idea is to use the history of device usage to construct features, that are used to provide a second factor for verifying an access request from a user with a claimed identity. Experiments in [13] showed that features extracted from device history can be effectively used to distinguish users. Although the approach is applicable to any computing device, it is primarily used to enhance security of mobile users carrying MIDs.

The user profile includes private information including (i) device data, such as GPS location data and WiFi/Bluetooth connections, (ii) carrier data, such as information on cell towers connected to the device, or Internet access pattern, and (iii) cloud data, such as calendar entries. The profile is stored at the carrier to ensure that a compromised device cannot be used to impersonate the legitimate user. This profile however includes private and potentially sensitive user data, that must be protected.

The aim of this paper is to address this problem: we propose an efficient method of privacy preserving implicit authentication systems, and model and prove its security.

We consider a network-based implicit authentication system where user authentication is performed collaboratively by the *device* (the MID), and the *carrier* (network service provider). *Application servers* will use the result of this authentication to grant access to users and do not directly participate in the authentication protocol.

Our implicit authentication protocol generates a score that will be used to accept to reject the user. The score is obtained through secure two party computation between the device and the carrier. User data is encrypted and stored at the carrier and is used by the interactive protocol to compute the authentication. Data privacy against a semi-honest carrier is guaranteed because the user data is stored in encrypted form. Calculating the score is by a specially designed secure two party computation protocol. Secure two party protocols can be implemented through general constructions using secure circuit evaluation, e.g. [27,11], or fully homomorphic encryption [9]. These general constructions however will be inefficient in practice.

Because no data is stored on the MID, user data stays protected even if the device is lost or stolen.

1.1 Our Contributions

The main contribution of this paper is proposing a profile matching function that uses the statistics of features to accept or reject a new sample presented by a user, and providing a privacy preserving protocol for computing a score function for a newly presented data. We assume the user profile is a vector of features, each corresponding to a random variable, $(V_1, \ldots, V_n)$, with an associated probability distribution. The distribution of V_i is stored as the set of values of the variables in the last ℓ_i successful logins. A new login attempt generates a vector of values, one for each feature. The verification function must decide if this vector indeed has been generated by the claimed user. Our proposed verification algorithm takes each feature separately and decides if the presented value is from the claimed user. The final verdict is reached by combining the decisions from all features. To determine if a new value presented for a feature v_i matches the model (stored distribution of the feature), we will use a statistical decision making approach that uses the *Average Absolute Deviation (AAD)* of the distribution. We use AAD to define an interval around the read value v_i given by $[v_i - AAD(V_i), v_i + AAD(V_i)]$ and determine the concentration of the stored values in the user profile that falls within this Interval: if the number is higher than a specified threshold, then the authentication algorithm accepts the reading. AAD and standard deviation are commonly used measures for estimating the spread of a distribution. Our verification algorithm effectively measures similarity of the presented value with "most common" readings of the variable. Using AAD allows more efficient private computation.

Constructing User Profiles. A user profile is a feature vector $(V_1, \ldots, V_n)$, where feature V_i is modelled by a vector of ℓ_i past samples. The vector can be seen as a sliding window that considers the latest ℓ_i successful authentication data. Using different ℓ_i is allowed for better estimation of the feature distribution. Possible features are the frequency of phone calls made or received by the user, user's typical locations at a particular time, commonly used WiFi access-points, websites that the user frequently visits, and the like. Some features might be dependent on other ones. For example, given that the user is in his office and it is lunch time, then there is a higher chance that he receives a call from home. We do not consider dependence of features and in selecting them make special care to select those that appear independent.

We note that usage data such as application accesses and past WiFi connections could enhance performance and usability of the device and applications. However to use such data securely as part of the authentication system, the data must be stored securely at the carrier to be protected from malicious parties who may get hold of the device. In practice a user can be given a choice to use certain device-collected data for authentication and so for such data.

Privacy-Preserving Authentication. All user profile data is stored in encrypted form on the carrier and the decryption keys are only known by the device. To find the authentication score for each feature, the device and the carrier have to perform a secure two-party computation that outputs the authentication score to the carrier, and nothing to the device. We propose a 3-round protocol between the device and the carrier that allows the carrier to "securely" calculate the score. To provide the required efficiency, we have to sacrifice some privacy in the sense that although the actual data samples are not leaked, the protocol does expose structural information related to the relative order of data samples. We give a formal definition of this notion of privacy which guarantees that no information other than the relative order of samples is revealed by a secure protocol. We then prove the security of the protocol, according to the definition, against semi-honest adversaries.

The paper is organized as follows. We discuss the related work in the field of behavioural authentication in Section 1.2. Section 2 contains the preliminaries needed for our scheme. System architecture, the adversarial model, and a basic implicit authentication protocol not guaranteeing privacy are presented in Section 3. We give details of our proposed protocols for semi-honest and malicious devices in Section 4. Security proofs and a detailed discussion on the efficiency of our proposed protocol are provided in the full version of this paper [21].

1.2 Related Work

The privacy problem in implicit authentication was noted in [13]. The three approaches proposed for enhancing privacy are: (i) removing unique identifier information; (ii) using pseudonyms; and (iii) using aggregate data instead of fine grained data. All these methods however have limited effectiveness in protecting users' privacy while maintaining usefulness of the system. It is well known that user data with identification information removed can be combined with other public data to re-identify individuals [24], and fixed pseudonyms does not prevent linkability of records [15]. Finally coarse aggregates result in inaccurate authentication decisions.

Further discussion on related work e.g. on privacy-preserving biometric systems [16,18,19,22] and implicit authentication systems using accelerometers [6], gait recognition [14], user location [5,23,7,4], and fingerprints [26] can be found in full version of this paper [21].

2 Preliminaries

Our constructions use homomorphic encryption and order preserving symmetric encryption. In the following we first give an overview of these primitives.

Homomorphic Encryption (HE). We use here an *additive homomorphic public key encryption scheme* [20,8] which supports addition and scalar multiplication in the ciphertext domain. Let $E^{HE}_{pk}(\cdot)$ denote such an encryption algorithm. Given encryptions of a and b, an encryption of $a+b$ can be computed as $E^{HE}_{pk}(a+b) = E^{HE}_{pk}(a) \odot E^{HE}_{pk}(b)$, where $\odot$ represents an efficient operation in the ciphertext space. The existence of the operation $\odot$ enables scalar multiplication to be possible in the ciphertext domain as well; that is, given an encryption of a, an encryption of ca can be calculated efficiently for a known c. To simplify the notation, we use $+$ for both the operations $+$ and $\odot$. As an instantiation, we use Paillier Cryptosystem [20,8] in which $E^{HE}_{pk}(x+y) = E^{HE}_{pk}(x)E^{HE}_{pk}(y)$ and $E^{HE}_{pk}(cx) = E^{HE}_{pk}(x)^c$. Paillier Cryptosystem is semantically secure under the decisional composite residuosity assumption [20,8].

Order Preserving Symmetric Encryption (OPSE). Order preserving symmetric encryption (OPSE) was introduced in [3]. A function $f : D \to R$ is order preserving if for all $i, j \in D$: $f(i) > f(j)$ if and only if $i > j$. A symmetric encryption scheme having plaintext and ciphertext space D, R is order preserving if its encryption algorithm is an order preserving function from D to R for all keys; i.e., an OPSE maps plaintext values to ciphertext space in such a way that the order of the plaintext values remains intact. The construction provided in [3] has been proven secure in the POPF-CCA (pseudorandom order preserving function against chosen-ciphertext attack) model. More details on the security model and encryption system are given in the full version of this paper [21].

Secure Two-Party Computation. In a secure two-party computation, two parties A and B with private inputs x and y, respectively, compute a function $f(x, y)$ ensuring that, *privacy* and *correctness* are guaranteed. Privacy means that neither A nor B learns anything about the other party's input. Correctness means that the output is indeed $f(x, y)$ and not something else. To formalize security of a two-party protocol, the execution of the protocol is compared to an "ideal execution" in which parties send their inputs to a trusted third party who computes the function using the inputs that it receives. Informally, a protocol is considered secure if a real adversary in a real execution can learn "the same" amount of information as, or can "change the protocol output" not more than what an ideal adversary can do in the ideal model.

Security of two-party protocols is considered against different types of adversaries. In the *semi-honest* model (a.k.a. honest-but-curious model), the adversary follows the protocol specification but tries to learn extra information from the protocol transcript. A *malicious* (a.k.a. dis-honest) adversary however follows an arbitrary strategy (bounded by polynomial time algorithms) and can deviate from the protocol specification.

There are a number of generic constructions for secure two party computation, e.g. [27,11], however they have proven to be too inefficient in practice, specially in resource-restricted devices. An alternative approach to realize specific secure two-party protocols is based on homomorphic encryption (HE). In this approach, one party sends its encrypted inputs to the other party, who then computes the specific desired function in the encrypted domain using the homomorphic properties of the encryption system. Paillier's additively homomorphic cryptosystem [20] and Gentry's fully homomorphic scheme [10] are the commonly used tools in this approach.

Average Absolute Deviation. In our protocol we use a model of feature comparison that uses average absolute deviation. The *median* of a data set is the numeric value separating the higher half of distribution from the lower half. The *average absolute deviation (AAD)* of a data set is the average of the absolute deviations and characterizes a summary of statistical dispersion of the data set. For a set $X = \{x_1, x_2, \ldots, x_N\}$ with a median denoted by $\mathrm{Med}(X)$, AAD is defined as $AAD(X) := \frac{1}{N} \sum_{i=1}^{N} |x_i - \mathrm{Med}(X)|$.

Notation. Throughout the paper we use E_{pk}^{HE} and D_{sk}^{HE} to denote the encryption and decryption algorithms of a homomorphic encryption scheme such as Paillier Cryptosystem with public and secret key pair (pk, sk). For the OPSE algorithm we use E_k^{OPSE} and D_k^{OPSE} to refer to the encryption and decryption with key k. Key generation algorithms are denoted by $KeyGen^{HE}$ and $KeyGen^{OPSE}$, respectively for HE and OPSE schemes.

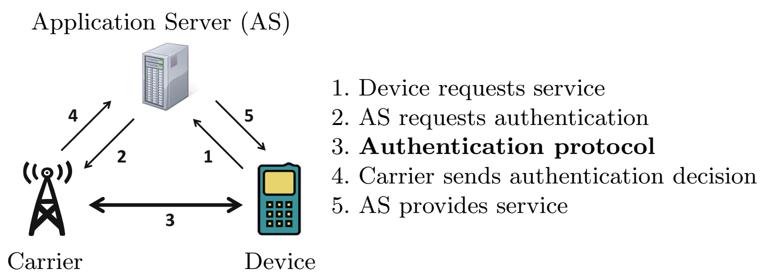
Application Server (AS)
4
5
2
1
3
Carrier
Device
1. Device requests service
2. AS requests authentication
3. **Authentication protocol**
4. Carrier sends authentication decision
5. AS provides service

Fig. 1. The System Architecture

3 System Model

Fig. 1 gives the working of the system we consider in practice. The authentication process is between the device and the carrier. In a typical scenario, an application server receives a service request from a device. The user information is forwarded to the carrier that will engage in the authentication protocol with the device. At the completion of the protocol, the results are sent to the application server, and if successful, the device (user) will receive the requested service.

The focus of this paper is on the device and carrier authentication. We assume other communication channels are secure and the information will be communicated safely across these channels. User data is stored in encrypted form at the carrier. The device records user's data, encrypts it and sends it to the carrier. No data used to develop the user profile in implicit authentication is stored on the device. This ensures that if the device is compromised, the adversary cannot learn the user profile and simulate its behaviour.

We only consider the data that is collected by the device to be included in the user profile. The information collected by the carrier is known to the carrier and is not included. Selection of an appropriate set of features that allow distinguishability of users is outside the scope of this paper. The goal of this paper is to provide privacy for user features that are used as part of the user's profile.

Trust Assumptions and the Adversarial Model. We assume that the carrier correctly follows the protocol but may try to learn the users' data. This is a reasonable assumption given the stature and reputation of carriers and difficulty of tracing the source of data leakage. We assume the device is used by the user for a period of time before being compromised. This is the period during which the user profile is constructed. We consider two types of adversaries. Firstly, we consider a less sophisticated adversary that tries to use a stolen device without tampering with the hardware or the software and so the device is assumed to follow the protocol. This also corresponds to the case that the authentication program resides in a tamper proof [12,17] part of the device and cannot be modified by the adversary and so a captured device follows the protocol but takes input data from the adversary. We assume the program can be read by the device holder, but cannot be changed. In the second case, the device behaves in a malicious way and may deviate from the protocol to succeed in an authentication scenario.

In both cases the system must guarantee privacy of the user: that is, neither the carrier nor the adversary in possession of the compromised device should learn the user's profile data. A stolen device used by an active malicious user must also fail in authentication.

3.1 Authentication without Privacy

A user profile consists of *features.* A feature is a random variable that can be sampled by the device and in combination with other features provides a reliable means of identifying users. We denote feature i by the random variable V_i that is sampled at each authentication request, and if the authentication is successful, is stored by the carrier and used as part of the distribution samples for evaluation of future authentication requests. The variable distribution for the i-th feature is approximated as $V_i = (v_i(t_1), v_i(t_2), \ldots, v_i(t_{l_i}))$. Here, $v_i(t_j)$ is the feature value at time t_j and l_i is a system parameter. As discussed before, we only consider independent features.

The *user profile* $\mathcal{U}$ is a tuple of features; that is $\mathcal{U} := (V_1, V_2, \ldots, V_n)$, where n is the total number of features. A *sampled feature vector* is denoted as $(v_1(t), v_2(t), \ldots, v_n(t))$ where $v_i(t)$ is the current instance of the variable V_i. Given a user profile and a new set of measured samples (for features), the scoring algorithm first calculates individual feature scores, and then combines them to generate a total score which is compared to a threshold. Authentication is considered successful if the score is higher than the threshold. The scoring algorithm works as follows.

We assume the final authentication score is obtained as a combination of authentication scores that are calculated for each feature separately. The *scoring function* for each variable estimates if the new sample belongs to the distribution that is represented by a set of samples from previous successful authentications. For a feature V_i we define our scoring function as follows:

$$s_i(t) = \Pr[\ b_l^i(t) \le V_i \le b_h^i(t)\ ], \text{ where}$$

$$b_l^i(t) = v_i(t) - \ \mathrm{AAD}(V_i) \quad \text{and} \quad b_h^i(t) = v_i(t) + \ \mathrm{AAD}(V_i)\ .$$

Here, $AAD(V_i)$ represents the average absolute deviation of data in the set V_i.

The probability $\Pr[\ b_l^i(t) \le V_i \le b_h^i(t)\ ]$ is approximated by counting the number of elements of V_i that fall between $b_l^i(t)$ and $b_h^i(t)$ and dividing the count by the number of all elements, i.e. l_i.

As will be shown in Section 4, the choice of $AAD(V_i)$ allows the carrier to perform the required computation on encrypted data. The scoring function estimates the likelihood of the new sample belonging to the distribution by counting the number of the previously recorded values of a feature that conform with, i.e. are within a determined interval of, the new sample.

To obtain the combined score of n features, various methods might be used depending on the authentication policy. A simple and popular method in this regard is the weighted sum of the scores as $a(t) := w_1 s_1(t) + \cdots + w_n s_n(t)$, where w_i represents the weight assigned to the i-th feature and $a(t)$ is the combined authentication score.

In summary, the authentication protocol proceeds as follows: The carrier has a user profile which consists of the sample distributions of the user features. The device sends a set of sampled behavioural data to the carrier. The carrier retrieves the sample distribution for each feature and calculates the feature scores. Then it combines the individual feature scores and compares the combined score with a threshold to make an authentication decision.

4 Privacy-Preserving Authentication

At the heart of the authentication protocol is the score computing algorithm. It basically takes two inputs: the stored distribution and the fresh device sample, and it produces a feature score. All the computation takes place at the carrier side, given the two inputs above, where the former is stored by the carrier, and the latter is provided by the device. Both inputs are in plaintext. In this section, we focus on this algorithm and provide a two-party score computing protocol that is able to calculate the feature score from *encrypted* profiles stored at the carrier and *encrypted* fresh samples provided by the device, where the keys to encryptions are only known to the device.

We chose to provide private protocols for score computation on the *feature score* level, as opposed to the *combined score* level, for two reasons: first, different carriers might have different authentication policies, and hence different score combination formulas, and our formulation choice leaves the choice of combination method open; second, we consider it an overkill to require that the carrier only finds out about the combined score and nothing about the individual scores, and indeed solutions for such an overkill are likely to be inefficient for practice.

In the following we propose a protocol between a device and a carrier that enables the carrier to calculate a feature score for the device, while provably guaranteeing that no information about the stored profile at the carrier is revealed to the device other than the AAD of the stored feature values, and no information about the fresh feature value provided by the device is revealed to the carrier other than how it is ordered with respect to the stored profile feature values.

4.1 A Protocol Secure against Honest-but-Curious Adversaries

Let $HE = (KeyGen^{HE}, E^{HE}, D^{HE})$ be a homomorphic encryption scheme, such as Paillier, and $OPSE = (KeyGen^{OPSE}, E^{OPSE}, D^{OPSE})$ be an order-preserving symmetric encryption scheme. The protocol Π_{PI} we propose consists of four phases: system setup, precomputation, authentication, and AAD update. The protocol works as follows:

System Setup. Performed once for each device, $KeyGen^{HE}$ and $KeyGen^{OPSE}$ are run to generate the HE key pair (pk, sk) and the OPSE key k_2. Public parameters of the two encryption systems HE and OPSE, including pk, are communicated to the carrier.

Precomputation. At any point during the life of the system, the carrier has stored an accumulated user profile containing $\left(E_{pk}^{HE}(v_i(t_j)), E_{k_2}^{OPSE}(v_i(t_j))\right)$ for $j = 1, \ldots, l_i$. Before the start of the authentication protocol, the carrier precomputes $E_{pk}^{HE}(AAD(V_i))$ as follows. It first computes:

$$E_{pk}^{HE}(\ AAD(V_i) \cdot l_i\ ) = \sum_{j=1}^{l_i} \left| E_{pk}^{HE}(v_i(t_j)) - E_{pk}^{HE}(\mathrm{Med}(V_i)) \right| ,$$

where $\mathrm{Med}(V_i)$ denotes the median element of V_i and can be found using the OPSE ciphertexts stored in the profile. Then the constant factor l_i is removed using the scalar multiplication property of the homomorphic encryption HE. In Paillier cryptosystem, this is done by raising $E_{pk}^{HE}(AAD(V_i) \cdot l_i)$ to the power of l_i^{-1}, where l_i^{-1} is precomputed once and stored along with l_i as system parameters.

Authentication. Device samples the features (i.e. user data) using its modules. For each feature value $v_i(t)$, $1 \leq i \leq n$, at time t, device computes a pair of encrypted values, $e_i(t) = (\ E_{pk}^{HE}(v_i(t)), E_{k_2}^{OPSE}(v_i(t))\ )$. The HE ciphertext allows the carrier to perform necessary computations, namely addition and scalar multiplication, in the encrypted domain, while the OPSE ciphertext helps the carrier find the order information necessary to the computation.

Device sends $e_i(t)$ values to the carrier. Using these values, carrier calculates $E_{pk}^{HE}(b_l^i(t))$ and $E_{pk}^{HE}(b_h^i(t))$ as follows:

$$E_{pk}^{HE}(b_l^i(t)) \leftarrow E_{pk}^{HE}(v_i(t)) - E_{pk}^{HE}(AAD(V_i))$$

$$E_{pk}^{HE}(b_h^i(t)) \leftarrow E_{pk}^{HE}(v_i(t)) + E_{pk}^{HE}(AAD(V_i))$$

where $V_i = \{v_i(t_1), \ldots, v_i(t_{l_i})\}$ and $E_{pk}^{HE}(AAD(V_i))$ is pre-computed as discussed.

Carrier however does not know the order of the newly generated encrypted values with respect to the stored ciphertexts in the user profile. To find the order, carrier interacts with the device: carrier first sends $E_{pk}^{HE}(b_l^i(t))$ and $E_{pk}^{HE}(b_h^i(t))$ (for all features) back to the device. Device decrypts the ciphertexts using the decryption function D_{sk}^{HE} and gets $b_l^i(t)$ and $b_h^i(t)$, and then encrypts the result to find $c_l^i(t) = E_{k_2}^{OPSE}(b_l^i(t))$ and $c_h^i(t) = E_{k_2}^{OPSE}(b_h^i(t))$, respectively, using the OPSE scheme. Device sends $c_l^i(t)$ and $c_h^i(t)$ back to the carrier.

Carrier computes the individual score $s_i(t)$ as the number of the OPSE ciphertexts $E_{k_2}^{OPSE}(V_i)$ in the profile that satisfy $c_l^i(t) \leq E_{k_2}^{OPSE}(V_i) \leq c_h^i(t)$. Note that this condition is equivalent to $b_l^i(t) \leq V_i \leq b_h^i(t)$.

The same process is used for all features. The final authentication score is then calculated using the score combination method, e.g. the weighted sum method described earlier. Finally, the final calculated score determines if implicit authentication is successful or not. If implicit authentication is not successful, the device is challenged on an explicit authentication method, e.g. the user is logged out of a service and prompted to log in anew by providing a password.

AAD Update. If implicit authentication is successful, or if it is unsuccessful however the device subsequently succeeds in explicitly authenticating itself, then the AAD needs to be updated using the new encrypted features provided in the authentication phase. The current feature history includes a vector of size l_i. The new feature is added to this vector first, and then, depending on the carrier's strategy, the oldest feature might be discarded to keep the vector size constant. In both cases, recalculating the AAD only needs constant-size differential calculations and there is no need to recompute AAD from scratch (which instead would be linear in the size of the vector). The reason is that when the median is shifted, for almost half of the existing feature values, the absolute deviation increases by the difference of the old and new medians, and for almost all of the rest of the existing feature values, the absolute deviation decreases by the same value, and these almost totally cancel each other out. Only a few calculations are needed eventually to account for the few that do not cancel out, plus the possible discarded feature, and the new feature.

Complexity. We discuss the computation complexity of the precomputation, authentication, and update phases of our protocol in the full version of this paper [21]. We implement Paillier and OPSE to confirm computation benchmarks in the literature, and calculate concrete running times for our protocol. In particular, we show that authentication takes *less than 300 milliseconds* on a typical device as a background process, and hence our protocol is able to protect user privacy with an insignificant computation overhead cost.

Security. We discuss the security of our protocol considering honest-but-curious devices and carriers in the full version of this paper [21]. We provide a formal definition of privacy for our protocol against honest-but-curious devices and carriers. The definition intuitively guarantees that by participating in the protocol, the device only learns the AAD of the usage data stored at the carrier side, and the carrier only learns little beyond the order information of the current sample with respect to the stored data. We argue that the AAD and order information learned during the protocol reveal little about the actual content of the data in question, and hence our definition guarantees a high level of privacy. Eventually, in the full version of this paper [21], we prove the following theorem guaranteeing the privacy of our protocol:

Theorem 1. *Protocol Π_{PI} is provably secure against honest-but-curious devices and honest-but-curious network carriers.*

4.2 Securing the Protocol against Malicious Devices

In the above version of the protocol, secure against honest but curious adversaries, in the authentication phase the carrier interacts with the device as follows:

the carrier sends homomorphic ciphertexts $E_{pk}^{HE}(b_l^i(t))$ and $E_{pk}^{HE}(b_h^i(t))$ to the device and the device is expected to reply back order-preserving ciphertexts of the same plaintexts, i.e. $E_{k_2}^{OPSE}(b_l^i(t))$ and $E_{k_2}^{OPSE}(b_h^i(t))$. These order-preserving ciphertexts are subsequently used to compare the values of $b_l^i(t)$ and $b_h^i(t)$ in the order-preserving ciphertext space with the feature values and find out how many feature values lie between $b_l^i(t)$ and $b_h^i(t)$. However, a malicious device cannot be trusted to return correctly formatted order-preserving ciphertexts. In the following, we propose a modified version of the protocol secure against malicious devices. We call this modified version $\Pi_{PI}*$.

First, we note that the device cannot be forced to use an honest feature value $v_i(t)$ to start with. In the absence of a trusted hardware such as tamper-proof hardware, the device may enter the interaction with the carrier on any arbitrary input. Even with the recent advances in smartphone technology, e.g. ARM's TrustZone [1], the device cannot be prevented to change the sensor readings unless the whole algorithm is run in the so called Trusted Execution Environment (TEE). However, the device can be required to show that the ciphertext $E_{pk}^{HE}(v_i(t))$ is well-formed. To enforce this requirement, we require that the device sends an interactive proof of knowledge of the corresponding plaintext $v_i(t)$ along with the ciphertext $E_{pk}^{HE}(v_i(t))$. Proofs of knowledge of plaintext exist for most public key encryption schemes. For Paillier encryption, a concrete proof protocol can be found in [2], which can be made non-interactive using the well-known Fiat-Shamir heuristic.

Apart from inclusion of the above proof of knowledge, further modification is required to make the protocol secure against malicious devices. The main idea here is as follows: instead of asking the device for order-preserving ciphertexts, the ability to interact with the device is used to directly compare $b_l^i(t)$ and $b_h^i(t)$ with the feature values, only using the homomorphic ciphertexts. In each round of interaction $b_l^i(t)$ (resp. $b_h^i(t)$) is compared with an element of the feature vector. The relative position of $b_l^i(t)$ (resp. $b_h^i(t)$) within the elements of the feature vector can be hence found in $\log l_i$ rounds of interaction following a binary search algorithm.

Assume that in one round the carrier wishes to compare $b_l^i(t)$ with $v_i(t_j)$. The carrier has homomorphic encryptions of both, i.e. $E_{pk}^{HE}(b_l^i(t))$ with $E_{pk}^{HE}(v_i(t_j))$, and hence can calculate $E_{pk}^{HE}(b_l^i(t) - v_i(t_j))$. The carrier is interested in knowing whether $b_l^i(t) - v_i(t_j)$ is positive, negative, or zero. The carrier chooses k random values and encrypts them using the homomorphic encryption scheme. It also randomises $E_{pk}^{HE}(b_l^i(t) - v_i(t_j))$ using scalar multiplication by either a positive or a negative random blinding factor. The carrier finally shuffles all the $k+1$ ciphertexts, including the k random cipheretxts and the blinded version of $E_{pk}^{HE}(b_l^i(t) - v_i(t_j))$, and sends them all to the device. The device decrypts all the received ciphertexts and replies back to the carrier with $k+1$ responses indicating whether each of the received ciphertexts decrypt to a positive, negative, or zero plaintext. The carrier knows what the response should be for the k random ciphertexts. Hence, it will first check whether all such responses are as expected.

If they are, then the carrier deducts whether $b_l^i(t) - v_i(t_j)$ is positive, negative, or zero, by reversing the effect of the random blinding factor.

The idea in the above interaction is that since all the $k+1$ challenges look random (and hence indistinguishable) to the device, a malicious device has at most $\frac{1}{k+1}$ chance of cheating and not getting caught. k is a parameter of the protocol and controls a trade-off between complexity and security. The larger k is, the less chance there is for a malicious device to cheat, but at the same time the higher the complexity of the protocol is.

Note that even if the device manages to cheat and not get caught, it does not gain any meaningful advantage in impersonating a legitimate user since the $b_l^i(t) - v_i(t_j)$ value is blinded before being sent to the device. Blinding changes the sign of the $b_l^i(t) - v_i(t_j)$ value with 50% probability. A malicious device therefore is not able to tell which behaviour, being honest or cheating, works in its favour.

Complexity. We discuss the computation complexity of the modified protocol in the full version of this paper [21]. In particular, we show that an authentication failure is discovered in *less than 4 seconds* after the first feature reading is reported by the device.

Security. We discuss the security of our protocol considering malicious devices in the full version of this paper [21]. We provide a formal definition of privacy for our protocol against maliciously-controlled devices. The definition intuitively guarantees that even if the device is maliciously controlled, it will not be able to learn any information more than what it would learn during an honest execution of the protocol. Eventually, in the full version of this paper [21], we prove the following theorem guaranteeing the privacy of our protocol:

Theorem 2. *Protocol $\Pi_{PI}*$ is provably secure against maliciously-controlled devices (with probability at least $\frac{k}{k+1}$), and is provably secure against honest-but-curious carriers.*

Conclusion

In this paper we proposed a privacy preserving implicit authentication system that can calculate authentication score using a realistic scoring function. We argued that using user behaviour as an additional factor in authentication has attractive applications. We showed that by relaxing the notion of privacy, one can construct efficient protocols that ensure user privacy and can be used in practice. The low computation and communication complexity of our proposed protocol in the case of semi-honest adversary makes it executable almost in real-time for carrier and modern MIDs. We also provided a modification to the basic protocol to ensure security in the case of a malicious device. Our proposed protocol in this case, has a complexity that grows logarithmically with the size of the user profile. We argued that this translates into a reasonable time-frame for implicit authentication with protection against malicious devices. A complete implementation of the system will be our future work.

References

1. ARM TrustZone, `http://www.arm.com/products/processors/technologies/trustzone`
2. Baudron, O., Fouque, P.-A., Pointcheval, D., Stern, J., Poupard, G.: Practical Multi-Candidate Election System. In: Proc. 20th ACM Symposium on Principles of Distributed Computing, pp. 274–283. ACM (2001)
3. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-Preserving Symmetric Encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 224–241. Springer, Heidelberg (2009)
4. Čapkun, S., Čagalj, M., Srivastava, M.: Secure Localization with Hidden and Mobile Base Stations. In: Int'l Conf. on Computer Communication, INFOCOM 2006 (2006)
5. Čapkun, S., Hubaux, J.-P.: Secure Positioning of Wireless Devices with Application to Sensor Networks. In: INFOCOM 2005: 24th Annual Joint Conf. of the IEEE Computer and Communications Societies, vol. 3, pp. 1917–1928. IEEE (2005)
6. Chang, K.-H., Hightower, J., Kveton, B.: Inferring Identity Using Accelerometers in Television Remote Controls. In: Tokuda, H., Beigl, M., Friday, A., Brush, A.J.B., Tobe, Y. (eds.) Pervasive 2009. LNCS, vol. 5538, pp. 151–167. Springer, Heidelberg (2009)
7. Chiang, J.T., Haas, J.J., Hu, Y.-C.: Secure and Precise Location Verification Using Distance Bounding and Simultaneous Multilateration. In: 2nd ACM Conference on Wireless Network Security, pp. 181–192. ACM (2009)
8. Damgård, I., Jurik, M.: Generalisation, A Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
9. Gentry, C.: A Fully Homomorphic Encryption Scheme. PhD thesis, Stanford University (2009)
10. Gentry, C., Halevi, S.: Implementing Gentry's Fully-Homomorphic Encryption Scheme. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011)
11. Goldreich, O., Micali, S., Wigderson, A.: How to Play Any Mental Game - A Completeness Theorem for Protocols with Honest Majority. In: Proc. 19th ACM Symposium on Theory of Computing, pp. 218–229. ACM (1987)
12. Haubert, E., Tucek, J., Brumbaugh, L., Yurcik, W.: Tamper-Resistant Storage Techniques for Multimedia Systems. In: Electronic Imaging 2005, pp. 30–40. International Society for Optics and Photonics (2005)
13. Jakobsson, M., Shi, E., Golle, P., Chow, R.: Implicit Authentication for Mobile Devices. In: Proc. of the 4th USENIX Conf. on Hot Topics in Security. USENIX Association (2009)
14. Kale, A., Rajagopalan, A., Cuntoor, N., Krüger, V.: Gait-Based Recognition of Humans Using Continuous HMMs. In: Proc. 5th IEEE Int'l Conf. on Automatic Face & Gesture Recognition, pp. 336–341. IEEE (2002)
15. Krumm, J.: Inference Attacks on Location Tracks. In: LaMarca, A., Langheinrich, M., Truong, K.N. (eds.) Pervasive 2007. LNCS, vol. 4480, pp. 127–143. Springer, Heidelberg (2007)
16. Leggett, J., Williams, G., Usnick, M., Longnecker, M.: Dynamic Identity Verification via Keystroke Characteristics. International Journal of Man-Machine Studies 35(6), 859–870 (1991)

17. Möller, S., Perlov, C., Jackson, W., Taussig, C., Forrest, S.R.: A Polymer Semiconductor Write-Once Read-Many-Times Memory. Nature 426(6963), 166–169 (2003)
18. Monrose, F., Rubin, A.: Authentication via Keystroke Dynamics. In: Proceedings of the 4th ACM Conference on Computer and Communications Security, pp. 48–56. ACM (1997)
19. Nisenson, M., Yariv, I., El-Yaniv, R., Meir, R.: Towards Behaviometric Security Systems: Learning to Identify a Typist. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 363–374. Springer, Heidelberg (2003)
20. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
21. Safa, N.A., Safavi-Naini, R., Shahandashti, S.F.: Privacy-Preserving Implicit Authentication. Cryptology ePrint Archive, Report 2014/203 (2014), `http://eprint.iacr.org/2014/203`
22. Shahandashti, S.F., Safavi-Naini, R., Ogunbona, P.: Private Fingerprint Matching. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 426–433. Springer, Heidelberg (2012)
23. Singelee, D., Preneel, B.: Location Verification Using Secure Distance Bounding Protocols. In: IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, pp. 840–846. IEEE (2005)
24. Tan, K., Yan, G., Yeo, J., Kotz, D.: A Correlation Attack Against User Mobility Privacy in a Large-Scale WLAN Network. In: Proc. of the 2010 ACM Workshop on Wireless of the Students, by the Students, for the Students, pp. 33–36. ACM (2010)
25. Tsai, C.-S., Lee, C.-C., Hwang, M.-S.: Password Authentication Schemes: Current Status and Key Issues. IJ Network Security 3(2), 101–115 (2006)
26. Wang, D.-S., Li, J.-P.: A New Fingerprint-Based Remote User Authentication Scheme Using Mobile Devices. In: Int'l Conf. on Apperceiving Computing and Intelligence Analysis (ICACIA 2009), pp. 65–68. IEEE (2009)
27. Yao, A.C.-C.: How to Generate and Exchange Secrets. In: 27th Annual Symposium on Foundations of Computer Science, pp. 162–167. IEEE (1986)