Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Efficient privacy-preserving implicit authentication

Alberto Blanco-Justicia, Josep Domingo-Ferrer*

Universitat Rovira i Virgili, Department of Computer Science and Mathematics, CYBERCAT-Center for Cybersecurity Research of Catalonia, UNESCO Chair in Data Privacy, Av Països Catalans 26, Tarragona, Catalonia E-43007

ARTICLE INFO

Keywords:
Privacy-preserving implicit authentication
Bloom filters
Privacy

ABSTRACT

The number of online service accounts per person has rapidly increased over the last years. Currently, people have tens to hundreds of online accounts on average, and it is clear that users do not choose new, different, and strong passwords for each of these accounts. On the other hand, it is quite inconvenient for the user to be forced to explicitly authenticate each time she wants to use one of her many accounts; this is especially true with small user devices like smartphones. Implicit authentication is a way to mitigate the preceding problems by authenticating individuals based not only on their identity and credentials, but on how they interact with a device, *i.e.* their behavior. User behavior can be characterized by collecting keystroke patterns, browser history and configuration, IP addresses and location, among other characteristics of the user. However, keeping the user's behavior profile in authentication servers can be viewed as privacy-invasive. Privacy-preserving implicit authentication has been recently introduced to protect the privacy of the users' profiles, specifically against the party performing the authentication, which we call the server in the sequel. Yet, the privacy-preserving implicit authentication schemes proposed so far involve substantial computation both by the user and the server. We propose here a practical mechanism based on comparing behavior feature sets encoded as Bloom filters. The new mechanism entails much less computation and can accommodate much more comprehensive sets of features than previous alternatives.

1. Introduction

Implicit authentication refers to a software system authenticating individuals based on the way they interact with their device, *i.e.* their behavior. In this context, the user behavior can be determined by collecting a variety of features, such as keystroke patterns, browser history and configuration, IP addresses, location, visible antennas, etc. Implicit authentication is a *complement*, rather than a substitute, of the usual explicit authentication based on identifiers and/or credentials. Authenticating implicitly can make life easier for users by reducing the number of times they have to authenticate explicitly.

Implicit authentication is gaining importance as the smartphone market rises. Relatively small and sometimes unwieldy screen keyboards in smartphones make typing strong passwords a difficult task. This situation, added to the well-known problem of weak password choices, repeatedly aired in the media, makes the use of secondary authentication mechanisms almost mandatory. Among these, biometric (fingerprint) authentication and two-factor authentication with one-time passwords are the most common choices. Biometric authentication has the shortcomings of needing special sensors in the user's device and requiring the authenticating server to acquire and store the user's reference biometric pattern. Two-factor authentication, on its side, has an intrinsic problem: the second channel (email, SMS, mobile app) used for confirmation is usually accessible on the same device (typically an Internet-enabled smartphone) used for the primary channel, so both channels may be simultaneously compromised.

Implicit authentication is not free of problems either. A salient issue is the privacy exposure of end users, who need to be profiled in order to provide a reference pattern against which their current behavior can be authenticated by the server. To keep the user's profile private against the server, researchers have proposed privacy-preserving implicit authentication [6,18]. In these proposals, the user's reference profile is stored in encrypted form at the server and the fresh usage sample captured by the user's device is compared against that reference profile. While this solves the privacy issues, it entails substantial computation, both by the user's device and the server.

1.1. Contribution and plan of this paper

We propose a computationally efficient privacy-preserving implicit authentication mechanism in which only fingerprints of the users' usage profiles are revealed. We use Bloom filters to encode the user's

[*] Corresponding author.
E-mail addresses: alberto.blanco@urv.cat (A. Blanco-Justicia), josep.domingo@urv.cat (J. Domingo-Ferrer).

https://doi.org/10.1016/j.comcom.2018.04.011
Received 16 November 2017; Received in revised form 9 March 2018; Accepted 13 April 2018
Available online 27 April 2018

reference profile and we leverage the properties of Bloom filters to compute the distance between the stored reference profile and the fresh samples provided by the user. The privacy of users' profiles is protected as long as cryptographic hash functions are secure.

Our proposed mechanism produces fingerprints of the feature sets that are compact and can be easily integrated in existing authentication protocols, for example, as headers in HTTP packets.

Section 2 gives an overview of related subjects, including implicit authentication systems and privacy-preserving implicit authentication systems. Section 3 describes the adversarial model. Section 4 describes Bloom filters in detail. Section 5 recalls the types of features included in users' profiles, and how to compute the dissimilarity between sets of features, depending on their type. Section 6 presents our proposed mechanism. Section 7 discusses the privacy and the security of our proposal. Finally, Section 8 analyzes the accuracy and the performance of the new system. Conclusions and future research lines are gathered in Section 9.

2. Related work

2.1. Implicit authentication

In implicit authentication, a server can authenticate users by checking whether their behavior is compatible or similar enough to their past recorded behavior. In this context, the user's behavior can be modeled as a combination of features like her browsing history, usual location, keystroke patterns, usually visible cell stations, etc. A user profile consists of one or several such sets of features.

In [11], empirical evidence was given that the features collected from the user's device history are effective to distinguish users and therefore can be used to implicitly authenticate them (instead of or in addition to explicit authentication based on the user's providing a password). Muncaster and Turk propose a general framework for continuous authentication of users in [16], based on the integration of active (*e.g.* fingerprints) and passive (*e.g.* keystroke patterns) biometric measurements. Dynamic Bayesian networks are used to aggregate the classification decisions and scores from the different biometric authentication mechanisms. They demonstrate the framework with face recognition and keystroke pattern analysis. Clarke and Furnell explore authentication using keystroke patterns analysis in [5]. The authors use neural networks to decide whether the users interacting with the smartphones' keyboards are the rightful owners of the devices. The authors acknowledge, however, that keystroke analysis is too dependent on the specific user (not all users use the keyboard enough, and some vary their writing patterns constantly), and so multimodal approaches seem a better solution. The authors in [19] propose SenGuard, an implicit authentication mechanism for mobile devices, which uses information from the touch screen, location, means of transport and voice patterns. Authentication decisions are taken with a space-time multi-modality classifier.

These proposals are designed as local authentication mechanisms to the mobile device and do not need to take privacy into account, since data are not meant to leave the local device. However, we argue that storing the profile in the user's device is insecure, because an intruder may gain access to it and learn sensitive information about the user, or even impersonate her. Therefore, it is safer to store the users' profiles in a secure facility, for example in the provider's premises. However, a user profile includes potentially sensitive data, and storing it outside the user's device violates her privacy.

This privacy risk is only partly mitigated by using a third party to store the users' profiles, for example the ISP or carrier. The typical architecture in this case consists of the user's device, a service provider and the carrier.

2.2. Privacy-preserving implicit authentication

In the privacy-preserving implicit authentication system proposed by Safa et al. [18], the user's device encrypts the user's usage profile at set-up time, and forwards it to the carrier, who stores it for later comparison. In this case, the security problem outlined in the previous section is fixed because the profile is never stored in the user's device (it is collected, encrypted, sent and immediately deleted by the device). Likewise, the privacy problem is also solved, because the profile sent to the carrier is encrypted. In fact, since the user's profile is exported in encrypted form, strictly speaking the carrier is no longer needed as a third party to store profiles and conduct the authentication: both functions could be performed by the service provider himself. Therefore, we will name the authenticating party as the server, which can be the carrier or the service provider.

The core of [18] is the algorithm for computing the dissimilarity score between two inputs: the fresh sample provided by the user's device and the profile stored at the server. All the computation takes place at the server and both inputs are encrypted: indeed, the server stores the encrypted profile and the user's device sends the *encrypted* fresh sample to the server. Note that the keys to both encryptions are only known to the user's device (it is the device that encrypts everything).

The server computes a dissimilarity score at the feature level, while provably guaranteeing that: (i) no information about the profile stored at the server is revealed to the device other than the average absolute deviation of the stored feature values; (ii) no information about the fresh feature value provided by the device is revealed to the server other than how it is ordered with respect to the stored profile feature values.

The score computation protocol uses two different encryption schemes: a homomorphic encryption scheme *HE* (for example, Paillier) and an order-preserving symmetric encryption scheme *OPSE*. This protocol is restricted to numerical features, due to the kind of computations that need to be performed on them. Such a limitation is a shortcoming, because behavior characterization may require non-numerical features, *e.g.* the browser history.

The implicit authentication system proposed by Domingo-Ferrer et al. [6] tries to overcome some of the limitations of the previous approach: it uses a single cryptosystem, it does not leak the order of fresh sample values, it does not leak the average absolute deviation of the stored feature values, and it can deal with non-numerical features. To do so, it builds on the work done by Blanco-Justicia et al. [2], which proposes a mechanism to compute the distance between preference functions, defined as sets of independent categorical features, correlated categorical features, or independent numerical features.

In [6], the user's device encodes and encrypts the user's profile using the Paillier cryptosystem and sends it to the server, along with some auxiliary values. Neither the encrypted set nor the auxiliary values reveal anything about the user's profile, except for the size of the set. Moreover, the user only keeps a secret auxiliary value and deletes the Paillier secret key, so the profiles cannot be recovered from the device either. To authenticate, the user's device sends an encrypted fresh sample of her activity to the server and then user and server engage in a two-party protocol to compute the distance between the stored or reference profile and the new sample. Note that the results are never decrypted, but checked using the server's and the user's auxiliary values.

Although Domingo-Ferrer et al. [6] solves several shortcomings of [18], it remains very demanding in computational terms. In fact, computing the auxiliary values during set-up requires inverting matrices, which in practice limits the maximum allowable size of the feature sets (due to computing time constraints).

In [22], the authors use a well-known approach in data mining, *i.e.* dimensionality reduction, to reduce the information revealed by the profiles, while keeping enough information for profiles to be correctly classified. This proposal, though, is limited to numerical features and