Reconciling User Privacy and Implicit Authentication for Mobile Devices*

Siamak F. Shahandashti
Newcastle University, UK
siamak.shahandashti@ncl.ac.uk

Reihaneh Safavi-Naini
University of Calgary, Canada
rei@ucalgary.ca

Nashad Ahmed Safa
University of Calgary, Canada

July 15, 2015

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 the 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 measurements of the features on the device, that will be subsequently matched against the features stored at the server, to accept or reject the request. The features however include private information such as user location or web sites that have been visited. We propose a *privacy-preserving implicit authentication* system that achieves implicit authentication without revealing information about the usage profiles of the users to the server. We propose an architecture, give a formal security model and a construction with provable security in two settings where: (i) the device follows the protocol, and (ii) the device is captured and behaves maliciously.

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

1 Introduction

In applications such as mobile commerce, users often provide authentication information using Mobile Internet Devices (MIDs) such as cell phones, tablets, and notebooks. In most cases, password is the primary method of authentication. The weaknesses of password-based authentication systems, including widespread use of weak passwords, have been widely studied (see e.g. [56] 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.

Two-factor authentication systems can potentially provide higher security. Second factors that use special hardware such as RSA SecurID tokens[1] or biometrics, incur additional cost which limit their wide usage. An attractive method of strengthening password systems is to use *implicit authentication* [30] as an additional factor for authentication. The idea is to use the history of a user's actions on the device, to construct a profile for the user consisting of a set of features, and employ it to verify a future authentication request. In the authentication phase, the device reports recent user behaviour, and authentication succeeds if the reported recent user behaviour "matches" her stored profile. Experiments in [30] showed that the features collected from the device history can be effectively used to distinguish users. Although the approach is general, it is primarily used to enhance security of mobile users carrying MIDs because of the richness of sensor and other data that can be collected on these devices. In such a scenario, a network service provider (the carrier) wishes to authenticate a user in possession of the MID.

An important distinction one needs to make is that the goal in implicit authentication is authenticating the user in possession of the device rather than the device itself. Consequently, the user profile needs to be stored at the carrier side to ensure that a compromised device cannot be used to impersonate the legitimate user.

The collected data about the user's actions can be divided into the following categories: (i) device data, such as GPS

*This manuscript has been accepted for publication in Computers & Security. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all disclaimers that apply to the journal apply to this manuscript. A definitive version is published in Computers & Security (2015) under DoI: 10.1016/j.cose.2015.05.009 [51].
This is an extended version of a paper that appeared in the proceedings of the 29th International Information Security and Privacy Conference IFIP SEC 2014 [49].

[1] www.emc.com/security/rsa-securid.htm

location data, WiFi/Bluetooth connections, and other sensor data, (ii) carrier data, such as information on cell towers seen by the device, or Internet access points, and (iii) third party data, such as cloud data, app usage data, and calendar entries. As discussed, the user profile including data from a mixture of these categories is stored at the carrier side. This profile however includes private and potentially sensitive user data, including device and third party data, that must be protected. One might be lead to think that there is an inherent trade-off between user privacy on one hand and the effectiveness of implicit authentication on the other. In this paper we show that this is a false trade-off; i.e., user privacy and effective implicit authentication can coexist. In particular, we propose an efficient privacy-preserving implicit authentication systems with verifiable 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), and will be used by *application servers* to authenticate users. The implicit authentication protocol generates a score for each feature representing the confidence level of the authentication based on that individual feature. Individual scores are subsequently combined based on the carrier authentication policy to accept or reject the user. Individual scores are obtained through a secure two party computation between the device and the carrier. Secure two party protocols can be constructed using generic constructions based on secure circuit evaluation, e.g. [60, 26], or fully homomorphic encryption [24]. We however opt to design a special-purpose protocol fit for the type of computations needed in implicit authentication. This allows us to achieve a level of efficiency which is practical and higher than those provided by generic constructions.

1.1 Our Contributions

We propose an implicit authentication system in which user data is encrypted and stored at the carrier, and an interactive protocol between the MID and the carrier is used to compute the authentication score. Data privacy is guaranteed since user data is stored in encrypted form. Because no data is stored on the MID, user data stays protected even if the device is lost or stolen. The main contributions of this paper are proposing a profile matching function that uses the statistics of features to accept or reject a new sample presented by a user, and designing a privacy-preserving protocol for computing a score function for newly presented data.

We assume the user profile is a vector of multiple features $(V_1, \ldots, V_n)$, each corresponding to a random variable with an associated probability distribution. Samples from 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 considers feature separately and computes a score for each feature indicating the confidence level in the presented value being from the claimed user. The final verdict is reached by combining the scores 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 reported value v_i given by $[v_i - \text{AAD}(V_i), v_i + \text{AAD}(V_i)]$ and then determine the score representing the concentration of past user behaviour observations close to the reported value v_i by counting the number of the stored values in the user profile that fall within the interval: the higher the number is the higher the score for that feature will be. Eventually the scores from all features are considered and the outcome of the authentication according to a certain policy is decided. AAD and standard deviation are commonly used statistical measures of dispersion, estimating the "spread" of a distribution. Our verification algorithm effectively measures similarity of the presented value with the "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. We survey the literature and find several features that are appropriate for our protocols. These are listed in Section 3.2. 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.

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 protocol that outputs the authentication score to the carrier, and nothing to the device. We propose two 3-round protocols between the device and the carrier that allow the carrier to "securely" calculate the score. The two protocols are designed to provide security in two different threat models where the device is considered either honest-but-curious or malicious. To provide the required efficiency, we have to sacrifice some privacy in the sense that although the actual data samples are not leaked, the protocols do expose limited structural information related to the relative order of data samples. We give formal definitions of our notions of privacy which guarantee that no information other than the relative order of samples is revealed by a secure protocol in the two threat models. We then prove the security of both protocols in the corresponding adversarial models.

The paper is organised 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 protocols. System architecture, adversarial models, and the core implicit authentication protocol not guaranteeing privacy are presented in Section 3. We give details of our proposed protocols for semi-honest and malicious devices and provide analyses on their computation and communication complexity in Section 4. Formal security definitions and proofs are provided in the appendices.

1.2 Related Work

The privacy problem in implicit authentication was noted in [30]. 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 [55], and fixed pseudonyms does not prevent linkability of records [35]. Finally coarse aggregates result in inaccurate authentication decisions.

User authentication is a widely studied problem with a wide range of mechanisms, ranging from cryptographic protocols to biometrics and systems that use special tokens [43, 8, 17, 2]. In a user study of authentication on mobile devices in [22], Furnell et al. showed that users prefer systems that authenticate the user periodically on a continuous basis throughout the day in order to maintain confidence in the identity of the user.

Implicit authentication systems that authenticate users continuously without disturbing the user, like the system considered in this paper, are the best fit for this requirement. These schemes have the potential to augment existing cryptographic and password-based authentication systems. Implicit authentication systems have been proposed based on many different features. These include systems that authenticate the user based on biometrics [36, 40, 42, 50], accelerometers [11], gait recognition [33, 23, 20], call and SMS pattern [52], location pattern [10, 53, 9, 12, 54], keystroke dynamics [14, 28, 61, 38, 19, 59], proximity to other devices [32, 48], and touchscreen dynamics [58, 16, 21]. Industry products such as Sentry from AdmitOne Security[2] and BehavioMobile from BehavioSec[3] implement implicit authentication. A major weakness of all these systems however is that the carrier learns the user's behaviour. Protecting the user's privacy in such a situation is the motivation for this paper.

Protecting user privacy in implicit authentication systems has been the motivating problem to a few works in the literature. The TrustCube framework proposed in [13] supports implicit authentication. The implicit authentication system based on keystroke dynamics in [41] also provides a level of privacy for the user. However, both of these systems require trusted remote platforms to carry out part or all of the computation. A trusted platform is typically implemented as a secure chip able to carry out limited security-sensitive operations. Examples of such platforms include the Trusted Platform Module (TPM), or its proposed mobile device equivalent a Mobile Trusted Module (MTM)[4]. Such trusted platforms are yet to be deployed extensively and we view this requirement as a limiting factor of these works. Finally, authors in [45] propose an implicit authentication system with user privacy that requires remote attestation, i.e., the calculations of the device being certified by a trusted third party. The involvement of such a trusted third party, we believe, significantly complicates and limits the usability of such a system. The aim of this paper is to achieve user privacy without requiring a trusted platform or involving a trusted third party.

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* [44, 15] 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 [44, 15] in which the operation $\odot$ can be carried out by multiplication of ciphertexts and scalar multiplication by exponentiation. Paillier cryptosystem is semantically secure under the decisional composite residuosity assumption [44, 15].

[2] www.admitonesecurity.com

[3] www.behaviosec.com

[4] Specifications of TPM and MTM are available from the Trusted Computing Group web page: www.trustedcomputinggroup.org

Order Preserving Encryption (OPE): A function $f : D \mapsto R$ is order preserving if for all $i, j \in D$: $f(i) > f(j)$ if and only if $i > j$. An encryption scheme with plaintext and ciphertext spaces D and R, respectively, is order preserving if its encryption algorithm is an order preserving function from D to R for all keys; i.e., an OPE maps plaintext values to ciphertext space in such a way that the order of the plaintext values remains intact. Order preserving (symmetric) encryption (OPE) was introduced in [5]. The provided construction was 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 Appendix A.

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 *correctness* and *privacy* are guaranteed. Correctness means that the output is indeed $f(x, y)$ and not something else. Privacy means that neither A nor B learns anything about the other party's input, other than what they would learn from the outputs (if any) that they receive by participating in the protocol. To formalise 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 communications. A *malicious* (a.k.a. dishonest) 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. [60, 26], however they have proven to be too inefficient in practice, specially in resource-restricted devices. An alternative approach to realise 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 [44] and Gentry's fully homomorphic scheme [25] 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 characterises a summary of statistical dispersion of the data set. For a set $X = \{x_1, x_2, \dots, x_N\}$ with a median denoted by $\text{Med}(X)$, AAD is defined as $\text{AAD}(X) = \frac{1}{N}\sum_{i=1}^{N} |x_i - \text{Med}(X)|$. Let N be an odd number and T and B denote respectively the set of top half and bottom half indexes, i.e., $T = \{i \mid x_i > \text{Med}(X)\}$ and $B = \{i \mid x_i < \text{Med}(X)\}$. Note that T and B are both of the same size. The above equation for calculating AAD can be simplified for an odd N as follows:

$$\text{AAD}(X) = \frac{1}{N}\Big(\sum_{i \in T} x_i - \sum_{i \in B} x_i \Big) . \qquad (1)$$

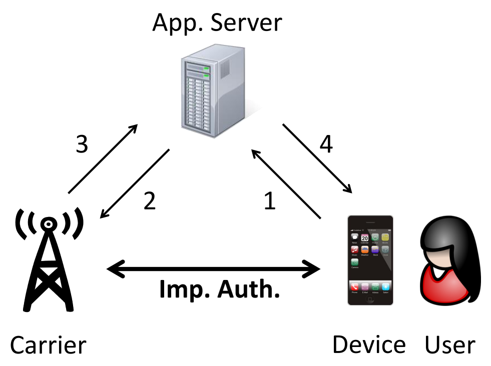
Notation: Throughout the paper we use E_{pk}^{HE} and D_{sk}^{HE} to denote the encryption and decryption algorithms of a homomorphic encryption scheme such as Paillier cryptosystem with public and secret key pair (pk, sk). For the OPE algorithm we use E_k^{OPE} and D_k^{OPE} to refer to the encryption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

3 System Model

We consider a system including three players: a device, a carrier, and an application server. A user who has the device wishes to obtain some service from the application server. The application server wishes to ensure that a legitimate user is in possession of the device, but at the same time does not want to require user's frequent active involvement to authenticate herself. Since the carrier is continuously providing service to the device, it has sufficient behavioural information on the user to be able to decide if the user in possession of the device is the legitimate user or not. Hence a natural course of action for the application server would be to consult with the carrier on the legitimacy of the user.

A typical protocol for the above scenario consists of the following sequence of messages. First the device requests the service from the application server, which subsequently sends a request to the carrier to pass its judgement on whether the user in possession of the device is the legitimate user or not. The carrier that has been continuously authenticating the user is then able to respond to the application server's request and the application server either provides or refuses service to the device accordingly. Figure 1 shows this scenario. Our focus in this paper is the continuous implicit authentication protocol between the carrier and the device. Note that this protocol runs continuously and is transparent to the user.

Throughout the paper we use "device" to refer to both the user and the device since the device is carrying out the



1. Device requests service
2. AS requests authentication
3. Carrier sends authentication decision
4. AS provides service

Figure 1: The system model and scenario. We propose a privacy-preserving *implicit authentication* protocol (denoted by Imp. Auth. above.), which is continuously carried out by the device and the carrier to determine whether the user in possession of the device is legitimate.

computations involved in the protocol. However note that the aim of implicit authentication is to authenticate the user in possession of the device.

Trust Assumptions and the Adversarial Model: We assume the communication channels in the protocol are secure and the information is communicated safely across these channels. User data is stored in encrypted form at the carrier. The device records user 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 her behaviour.

We aim to protect the data collected by the device, and thus in our protocol we only consider such device data. The information collected by the carrier is known to the carrier and is not included. Selection of an appropriate set of features that allow sufficient distinguishability of users is outside the scope of this paper. The goal here is to provide privacy for user features that are used as part of the user profile. Nevertheless, we give concrete examples of such distinguishing features from the literature in Section 3.2.

We assume that the carrier correctly follows the protocol but tries to learn user data through the information it receives by participating in the implicit authentication protocol. This, we believe, is a reasonable assumption given the stature and reputation of carriers on one hand, and the difficulty of tracing the source of data leakage on the other. We assume the device is used by the legitimate 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 [27, 39] 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 arbitrarily to succeed in the authentication protocol. This corresponds to the case where the device software or hardware is tampered with by the adversary in possession of the device.

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 profile data. Naturally, a stolen device used by an illegitimate user must also fail in authentication.

3.1 Authentication without Privacy

A user profile consists of a record of the distributions of one or more behavioural *features* of the user. A feature is a random variable that can be sampled by the device and in combination with other features provides a reliable means of distinguishing users. We denote feature i by the random variable V_i that is sampled at each authentication request. If the authentication is successful, the sample 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 $V_i = (v_i(t_1), v_i(t_2), \ldots, v_i(t_{\ell_i}))$. Here, $v_i(t_j)$ is the feature value at time t_j and ℓ_i is the length of the feature vector stored in the profile. This feature vector length is a system parameter which is assumed to be an odd integer to simplify the calculation of the mean. As discussed before, we only consider independent features.

Let us denote the user profile by $\mathcal{U}$. The profile consists of a tuple of n features; that is $\mathcal{U} = (V_1, V_2, \ldots, V_n)$.

The implicit authentication protocol is carried out between a carrier and a device. The protocol is carried out continuously and periodically. We consider one round of authentication in the following. At the beginning of each round, the carrier is in possession of a user profile $\mathcal{U}$ that consists of features V_i. The device wishes to authenticate itself to the carrier as a device whose user behaviour matches the recorded user profile $\mathcal{U}$. The protocol works as follows. The device samples the current features $\{v_i(t)\}_{i=1}^{n}$ and reports them to the carrier. The carrier considers each reported feature sample $v_i(t)$ and by comparing it to the sample distribution $V_i = (v_i(t_1), v_i(t_2), \ldots, v_i(t_{\ell_i}))$ from the user profile, decide how likely it is that the reported sample belongs to this distribution. We call this likelihood the

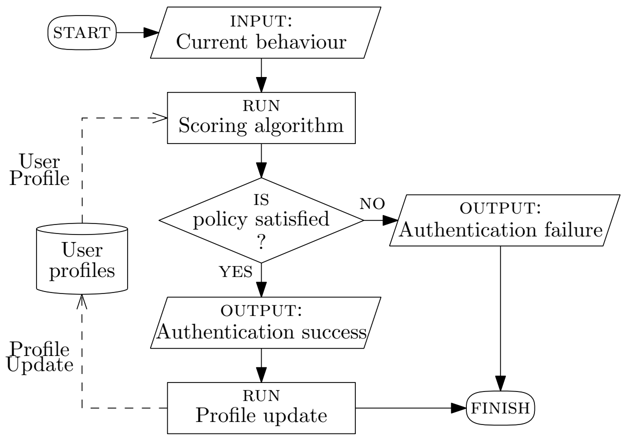


Figure 2: The authentication protocol flow in each round. Data flow is denoted by dashed arrows.

authentication *score* for the feature i, and denote it by s_i. Having calculated all the individual feature scores $\{s_i\}_{i=1}^n$, the carrier then decides based on a policy if the authentication succeeds or not. At the end of the round, if the authentication succeeds, the carrier updates the user profile to include the reported sample in the recorded samples. Figure 2 shows the flow of the authentication protocol in each round.

The authentication policy may vary between different carriers and it is crucial for an implicit authentication protocol to be able to support various carrier authentication policies. An example of a policy is one that requires each score to be above a certain threshold. Another carrier might require that at least a certain number of feature scores are above their corresponding threshold values. A simple and popular authentication policy is to require that a weighted linear combination of feature scores is above a certain threshold. In this case, the feature scores are combined linearly to calculate a combined score as $S = w_1 s_1(t) + \cdots + w_n s_n(t)$, where w_i represents the weight assigned to the i-th feature and S is the combined authentication score.

Each individual feature score is calculated by the carrier as the likelihood that a reported feature value belongs to the corresponding sample distribution recorded as part of the user profile. In this paper, we propose a simple and effective method for calculating these scores as follows. The carrier first calculates a measure of dispersion for the recorded sample distribution, namely the average absolute deviation (AAD). Then, the carrier considers an interval centred around the reported sample, with a length double the size of the AAD. The carrier counts the number of recorded samples from the user profile that fall within the above interval and considers the proportion of recorded samples that fall within the interval to be the score for the feature. Intuitively, the closer the reported sample is to the centre of concentration for the recorded samples, the more recorded samples will fall in the above interval, and hence the higher the feature score will be.

More formally, let $\mathrm{AAD}(V_i)$ represent the average absolute deviation of data in the set V_i. Also let the reported value for the i-th feature at time t be denoted by $v_i(t)$. For a feature V_i we define our scoring function at time t as follows:

$$s_i(t) = \Pr[\ b_i^{\mathrm{L}}(t) \le V_i \le b_i^{\mathrm{H}}(t)\ ],\ \text{where} \tag{2}$$

$$b_i^{\mathrm{L}}(t) = v_i(t) - \mathrm{AAD}(V_i) \quad \text{and} \quad b_i^{\mathrm{H}}(t) = v_i(t) + \mathrm{AAD}(V_i)\ .$$

The probability $\Pr[\ b_i^{\mathrm{L}}(t) \le V_i \le b_i^{\mathrm{H}}(t)\ ]$ is approximated by counting the number of elements of V_i that fall within the interval $[b_i^{\mathrm{L}}(t), b_i^{\mathrm{H}}(t)]$ and dividing the count by the number of all elements, i.e. ℓ_i, thus calculating the proportion of elements that fall within the interval.

The above method can in theory work with any reasonable measure of dispersion instead of AAD. However, as will be shown in Section 4, the choice of $\mathrm{AAD}(V_i)$ allows the carrier to perform the required computation on encrypted data.

3.2 Feature Selection

There are many works in the literature on distinguishing users through their different usage patterns. Most of these works use multiple features extracted from different sources of measurement and then export all the extracted features to a server in which a machine learning algorithm is first trained and then employed to tell users apart. Since such solutions do not address the issue of privacy, they can afford to use sophisticated algorithms based on arbitrarily chosen features which might not each be sufficiently discriminating on their own between different usage patterns. Our authentication protocols, on the other hand, require features that are reasonably discriminating on their own. A more careful look at the literature reveals that many such features are indeed available. In the following we list some candidates to be used as features in our protocols. Note that all the following candidates can be categorised under "device data", as described in Section 1.

Perhaps the most natural choice is the device location as sensed by GPS sensors. Jakobsson et al. analysed the location traces for participants in their study and found that they tend to be concentrated in three clusters corresponding to where the user lives, works, and shops [30]. Furthermore, user's location is highly correlated with the time of day and day of week. Hence, device latitude and longitude at specific times of day and days of week make good choices as features in our system. Effectively, implicit authentication using device location will then succeed with a good probability if the device is being used in a "usual" location, and fail with a good probability otherwise.

The study by Kang et al. [34] provides a few other feature candidates. They investigate smartphone usage patterns and their results show for example that although average daily device idle time does not vary much amongst different users, power consumption while idle as a percentage of total power consumption by the device varies significantly between different users and hence may be considered a distinguishing factor. They also find out that WiFi session durations for different users are concentrated around considerably different average values which span about an order of magnitude of varying lengths. Perhaps more interestingly, their results also demonstrate that users exhibit different and distinct habits in terms of when they start charging their smartphone. The median battery level at the start of charging varies from around 20% for some users to around 80% for others. Users with usually lower battery levels at the start of charging are the ones who are comfortable to wait until their battery is quite drained before worrying about recharging, whereas those with usually higher battery levels at the start of charging actively ensure that they have a good amount of charge on their battery most of the time.

Another interesting study is that of Falaki et al. [18]. They show among other results that users spend quite different and distinctive amounts of time interacting with their smartphones during the period roughly corresponding to "normal working hours" of 9-to-5 (i.e., 9:00 to 17:00), whereas the patterns of their interaction times may not be as significantly distinguishable during other hours of day or the weekend. This 9-to-5 interaction time is distributed around an average which can vary between around 10 minutes per hour to around 20 minutes per hour.

As a concrete system example, consider one that has the following features: device location latitude and longitude, power consumption while idle (henceforth PCI), WiFi session length (henceforth WSL), and battery level at the start of charging (henceforth BLS). Let all the features be reported on an hourly basis, with latitude and longitude being the GPS reading at the time, BCI being the power consumption while idle in the past hour as a percentage of the total power consumption in the past hour, WSL being the total WiFi session length in minutes in the past hour, and BLS being reported in percentage and only present if charging has started in the past hour. A possible implicit authentication policy may be as follows: scores from latitude and longitude are first considered; if they are both above certain thresholds, then at least one of the other scores, i.e., scores from PCI, WSL, and possibly BLS, needs to be above a certain threshold for implicit authentication to succeed; otherwise, all of the other scores need to be above certain thresholds for implicit authentication to succeed. Effectively, in such a system if the device is located where it is usually used, implicit authentication succeeds if the usage pattern (expressed as PCI, WSL, and BLS) loosely follows the previous usage pattern of the device, and if the device is located somewhere else, the usage pattern must strictly conform to the previous usage pattern for implicit authentication to succeed.

For all of the features mentioned above, the reported usage pattern seems to be highly dependent on the time of day and day of week. Hence, it would make sense to compare a reported feature by the device only to those in the recorded usage history profile that belong to the same time of day and day of week. That is, usage pattern reported on a Wednesday at 17:00 would only be compared to usage pattern history on previous Wednesdays at around the same time.

Note that although we have focussed on measurements of continuous variables (such as latitude and longitude, amount of power, and duration of time) in our examples above, we expect that our proposed protocols would work just as well for any discrete variable or in fact for any ordered nominal data type. As an example of a discrete variable, the 9-to-5 interaction time discussed above may be replaced by the number of sessions of interactions during 9-to-5, which as Falaki et al. show is also distinct between different users [18]. As an example of an ordered nominal variable, the activity classes that the Jigsaw sensing engine [37] provides may be considered as a feature in our system. Using the accelerometer sensor, Jigsaw is able to robustly classify user's activity as being stationary, walking, running, cycling, and in a moving vehicle. Assuming users have certain distinctive routines of for example cycling to work around a certain hour of day, the activity output by Jigsaw in different hours of day can potentially constitute features that can reasonably distinguish between different user behaviours. In some cases, other scoring functions such as those suggested by Jakobsson et al. [30], e.g. estimating $\Pr[V_i = v_i]$ or $\Pr[V_i \geq v_i]$ instead of what we propose (see Equation 2), would be more appropriate. Our protocols are generic in the sense that they can be easily modified to support such variants.

4 Privacy-Preserving Authentication

At the heart of the authentication protocol proposed in the previous section 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 decryption keys 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 our formulation leaves the choice of a specific authentication policy open for the carrier; second, we consider it an overkill to require that the carrier only finds out a potential combined score and nothing about the individual scores, and indeed solutions satisfying such a requirement are likely to be inefficient in 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 Semi-Honest Adversaries

Let $HE = (KeyGen^{HE}, E^{HE}, D^{HE})$ be a homomorphic encryption scheme, such as the Paillier cryptosystem, and $OPE = (KeyGen^{OPE}, E^{OPE}, D^{OPE})$ be an order-preserving encryption scheme. The protocol Π we propose consists of four phases: system setup, (user) profile initialisation, authentication, and profile update. System setup and profile initialisation are carried out once per device, but afterwards the authentication and profile update phases are carried out once per authentication round. Authentication rounds are carried out periodically and continuously. The protocol works as follows:

Phase 1. System Setup: Performed once for each device, $KeyGen^{HE}$ and $KeyGen^{OPE}$ are run by the device to generate the HE key pair (pk, sk) and the OPE key k_2. Public parameters of the two encryption systems HE and OPE, including pk, are communicated to the carrier.

Phase 2. Profile Initialisation: This phase is performed only once for each device to record the initial ℓ_i feature readings and compute an initial AAD for each feature. During this phase the device is assumed to be honest. Recall that implicit authentication requires a period of honest device usage to set up a usage profile based on which it is subsequently able to authenticate the user. During this phase, the device periodically sends HE and OPE encrypted feature readings $e_i(t) = E^{HE}_{pk}(v_i(t))$ and $e'_i(t) = E^{OPE}_{k_2}(v_i(t))$ to the carrier. The communications end after ℓ_i feature readings. At the end of this phase, the carrier has $2\ell_i$ ciphertexts for the i-th feature: $\{\ e_i(t_j), e'_i(t_j)\ \}_{j=1}^{\ell_i}$. Since the OPE ciphertexts $e'_i(t_j)$ enable the carrier to compare the corresponding plaintexts, the carrier is able to find the HE encryption of the median of the feature readings $E^{HE}_{pk}(\mathrm{Med}(V_i))$, where $\mathrm{Med}(V_i)$ denotes the median of $\{\ v_i(t_j)\ \}_{j=1}^{\ell_i}$. The carrier finds the indexes of the top and bottom half of plaintexts with respect to the median. Let us denote the set of top half indexes by T_i and the set of bottom half indexes by B_i. In other words:

$$T_i = \{j | v_i(t_j) > \mathrm{Med}(V_i)\}, \quad B_i = \{j | v_i(t_j) < \mathrm{Med}(V_i)\}.$$

Now the carrier uses the homomorphic property of HE to compute the encryption of AAD based on Equation 1 as follows:

$$E^{HE}_{pk}(\mathrm{AAD}(V_i)) = \ell_i^{-1} \cdot \Big(\sum_{j \in T_i} e_i(t_j) - \sum_{j \in B_i} e_i(t_j) \Big).$$

The setup and profile initialisation phases are now complete and from now on, the system will enter the mode in which the device is not trusted any more. In this mode, a continuous and periodic succession of authentication and profile update phases will be carried out.

Phase 3. Authentication: The device and the carrier enter the authentication phase with the carrier holding a profile of the device user including ℓ_i HE ciphertexts for the i-th feature: $\{\ e_i(t_j) = E^{HE}_{pk}(v_i(t_j))\ \}_{j=1}^{\ell_i}$, the ℓ_i corresponding OPE ciphertexts for the i-th feature: $\{\ e'_i(t_j) = E^{OPE}_{k_2}(v_i(t_j))\ \}_{j=1}^{\ell_i}$, and the HE encryption of the AAD of the features $E^{HE}_{pk}(\mathrm{AAD}(V_i))$. The device reports to the carrier the encryptions of a new reading as follows:

$$e_i(t) = E^{HE}_{pk}(v_i(t)) \quad \text{and} \quad e'_i(t) = E^{OPE}_{k_2}(v_i(t)) \ .$$

The HE ciphertext allows the carrier to perform necessary computations, namely addition and scalar multiplication, in the encrypted domain, while the OPE ciphertext helps the carrier find the order information necessary to the computation. The carrier calculates $E^{HE}_{pk}(b^{\mathrm{L}}_i(t))$ and $E^{HE}_{pk}(b^{\mathrm{H}}_i(t))$ as follows:

$$E^{HE}_{pk}(b^{\mathrm{L}}_i(t)) \leftarrow E^{HE}_{pk}(v_i(t)) - E^{HE}_{pk}(\mathrm{AAD}(V_i)) \ ,$$

$$E^{HE}_{pk}(b^{\mathrm{H}}_i(t)) \leftarrow E^{HE}_{pk}(v_i(t)) + E^{HE}_{pk}(\mathrm{AAD}(V_i)) \ .$$

The 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, the carrier interacts with the device as follows: the carrier first sends $E^{HE}_{pk}(b^{\mathrm{L}}_i(t))$ and $E^{HE}_{pk}(b^{\mathrm{H}}_i(t))$ back to the device for all features. The device decrypts the ciphertexts using the decryption function D^{HE}_{sk} and obtains $b^{\mathrm{L}}_i(t)$ and $b^{\mathrm{H}}_i(t)$, and then encrypts them to compute the following OPE ciphertexts:

$$c^{\mathrm{L}}_i(t) = E^{OPE}_{k_2}(b^{\mathrm{L}}_i(t)) \quad \text{and} \quad c^{\mathrm{H}}_i(t) = E^{OPE}_{k_2}(b^{\mathrm{H}}_i(t)) \ .$$

The device sends $c^{\mathrm{L}}_i(t)$ and $c^{\mathrm{H}}_i(t)$ back to the carrier. The carrier computes the individual score $s_i(t)$ as the number of the OPE ciphertexts $e'_i(t_j)$ in the profile that satisfy $c^{\mathrm{L}}_i(t) \leq e'_i(t_j) \leq c^{\mathrm{H}}_i(t)$. Note that this condition is equivalent to $b^{\mathrm{L}}_i(t) \leq v_i(t_j) \leq b^{\mathrm{H}}_i(t)$. Note that the scores are all calculated in parallel, and in only three rounds of interaction. The final authentication decision is then made by the

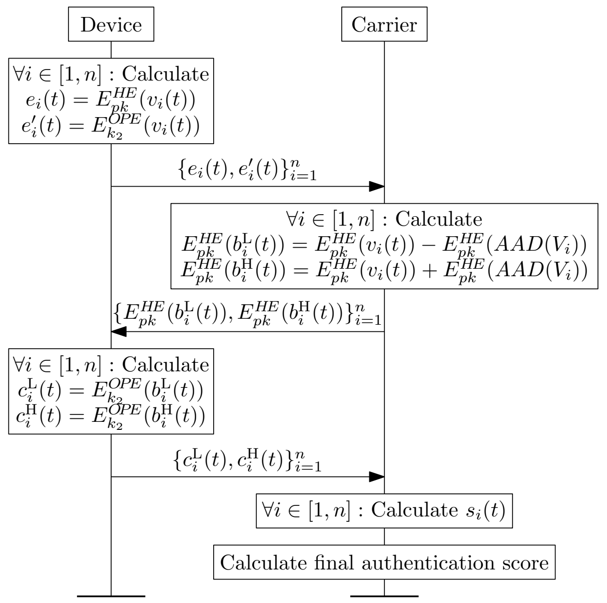

Device
Carrier
$\forall i \in [1,n]$: Calculate
$e_i(t) = E_{pk}^{HE}(v_i(t))$
$e'_i(t) = E_{k_2}^{OPE}(v_i(t))$
$\{e_i(t), e'_i(t)\}_{i=1}^n$
$\forall i \in [1,n]$: Calculate
$E_{pk}^{HE}(b_i^{\mathrm{L}}(t)) = E_{pk}^{HE}(v_i(t)) - E_{pk}^{HE}(AAD(V_i))$
$E_{pk}^{HE}(b_i^{\mathrm{H}}(t)) = E_{pk}^{HE}(v_i(t)) + E_{pk}^{HE}(AAD(V_i))$
$\{E_{pk}^{HE}(b_i^{\mathrm{L}}(t)), E_{pk}^{HE}(b_i^{\mathrm{H}}(t))\}_{i=1}^n$
$\forall i \in [1,n]$: Calculate
$c_i^{\mathrm{L}}(t) = E_{k_2}^{OPE}(b_i^{\mathrm{L}}(t))$
$c_i^{\mathrm{H}}(t) = E_{k_2}^{OPE}(b_i^{\mathrm{H}}(t))$
$\{c_i^{\mathrm{L}}(t), c_i^{\mathrm{H}}(t)\}_{i=1}^n$
$\forall i \in [1,n]$: Calculate $s_i(t)$
Calculate final authentication score

Figure 3: The authentication phase of our protocol Π

carrier based on its authentication policy, e.g. the weighted sum method described earlier in Section 3.1. 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. If either implicit or explicit authentication is successful, the carrier enters the profile update phase. Figure 3 shows the interaction diagram of the authentication phase of the protocol.

Phase 4. Profile Update: The carrier enters this phase after a successful implicit or explicit authentication. The carrier has the ciphertext for a new feature value $e_i(t) = E_{pk}^{HE}(v_i(t))$, and from the authentication phase it knows how $v_i(t)$ compares with the previously recorded features $\{v_i(t_j)\}_{j=1}^{\ell_i}$. The carrier updates the recorded features and the AAD as follows. Assume $e_i(t_1)$ is the ciphertext corresponding to the oldest feature and it is to be omitted from the feature list, and instead the new feature ciphertext $e_i(t_{\ell_i+1}) = e_i(t)$ added. Let T_i^{old} and B_i^{old} respectively denote the set of top and bottom half indexes for the old features $\{v_i(t_j)\}_{j=1}^{\ell_i}$, and T_i^{new} and B_i^{new} denote sets defined similarly for the updated features $\{v_i(t_j)\}_{j=2}^{\ell_i+1}$. Also let $\text{AAD}^{\text{old}}(V_i)$ and $\text{AAD}^{\text{new}}(V_i)$ denote the old and updated AADs. We have

$$E_{pk}^{HE}(\text{AAD}^{\text{old}}(V_i)) = \ell_i^{-1} \cdot \Big(\sum_{j \in T_i^{\text{old}}} e_i(t_j) - \sum_{j \in B_i^{\text{old}}} e_i(t_j) \Big),$$

$$E_{pk}^{HE}(\text{AAD}^{\text{new}}(V_i)) = \ell_i^{-1} \cdot \Big(\sum_{j \in T_i^{\text{new}}} e_i(t_j) - \sum_{j \in B_i^{\text{new}}} e_i(t_j) \Big).$$

Let us denote by Δ_i the difference between the two AAD ciphertexts times ℓ_i, i.e.

$$\Delta_i = \ell_i \cdot (\ E_{pk}^{HE}(\text{AAD}^{\text{new}}(V_i)) - E_{pk}^{HE}(\text{AAD}^{\text{old}}(V_i))\ ).$$

If Δ_i is calculated, the updated AAD can be calculated as follows:

$$E_{pk}^{HE}(\text{AAD}^{\text{new}}(V_i)) = E_{pk}^{HE}(\text{AAD}^{\text{old}}(V_i)) + \ell_i^{-1} \cdot \Delta_i.$$

Let $\setminus$ denote the set difference operation. To calculate Δ_i given T_i^{old}, B_i^{old}, T_i^{new}, and B_i^{new}, the carrier computes the following:

$$\begin{aligned} \Delta_i \ = \ & \sum_{j \in T_i^{\text{new}} \setminus T_i^{\text{old}}} e_i(t_j) \ - \sum_{j \in T_i^{\text{old}} \setminus T_i^{\text{new}}} e_i(t_j) \\ & - \sum_{j \in B_i^{\text{new}} \setminus T_i^{\text{old}}} e_i(t_j) \ + \sum_{j \in B_i^{\text{old}} \setminus T_i^{\text{new}}} e_i(t_j)\ . \end{aligned}$$

Note that each of the above four set differences includes at most one element. This means the profile update phase can be carried out very efficiently. At the end of this phase, the carrier holds a set of updated feature ciphertexts and an updated AAD ciphertext. The carrier will enter the authentication phase afterwards and wait for a new feature reading to be reported from the device.

Complexity: We discuss the computation complexity of the profile initialisation, authentication, and profile update phases of our protocol Π in the following. We also implemented Paillier and OPE to confirm computation benchmarks in the literature, and calculate concrete running times for our protocol. In the following we analyse the computation complexity of the protocol for one feature. To calculate approximate execution times for multiple features, the figures may be multiplied by the number of features.

In the profile initialisation phase, the device calculates a total of ℓ_i HE encryptions and ℓ_i OPE encryptions, and the carrier calculates ℓ_i ciphertext-space homomorphic additions and 1 ciphertext-space homomorphic scalar multiplication. Recall that this phase is only executed once.

The computation in the authentication phase is dominated by 1 homomorphic encryption, 2 homomorphic decryptions, and 3 order-preserving encryptions on the device side, and 2 ciphertext-space homomorphic additions (implemented in Paillier scheme by multiplications) on the carrier side, for each feature.

In the profile update phase, the carrier performs 4 ciphertext-space homomorphic additions and 1 ciphertext-space homomorphic scalar multiplication.

For typical parameters and on platforms comparable to today's smart-phones, HE encryption, decryption, and OPE encryption and decryption each take at most in the order of a few tens of milliseconds, as reported by previous works on the implementation of Paillier, and recent works on the implementation of OPE. We confirm these benchmarks through implementations of our own. Hence, we can see that the authentication phase for one feature is almost real-time. For multiple features, this phase can take at the longest in the order of a second to complete, which is reasonable given that there is no requirement for implicit authentication to be real-time.

To confirm the efficiency of Paillier homomorphic encryption and OPE implementations, we have benchmarked both schemes using Java-based implementations for both on an Intel 2.66 GHz core 2 duo processor (which is comparable to the processors of today's smartphones) while running other processes (including web browser, word processor, terminal, music player etc.) in the background. Hyper threading was not activated and only one core was used by the implementation.

For Paillier with 1024-bit keys (i.e. moduli) we have found that encryption takes 26 ms (milliseconds), decryption takes 35 ms, and both homomorphic addition and homomorphic scalar multiplication take negligible time comparatively (both are more than 500 times faster). We did not apply any optimisation techniques. These benchmarks are comparable to the ones reported in the literature on PCs with comparable specifications. Basu, Kikuchi, and Vaidya report comparable results, namely encryption times of 17 and 125 ms, and decryption times of 17 and 124 ms, with 1024-bit and 2048-bit keys, respectively [3]. Jakobsen, Makkes, and Nielsen report encryption times of 8 and 17 ms for optimised Paillier with 1024-bit and 2048-bit keys, respectively [29]. We use our own 1024-bit Paillier benchmarks, i.e. HE encryption: 26 ms, HE decryption: 35 ms, to demonstrate how efficient a simple implementation of our protocol can be in practice. Although, from the above examples it can be seen that optimised versions of the protocol can achieve higher efficiency even when 2048-bit keys are used. To provide some insight into how efficiency can be improved using optimisation techniques, we also give concrete execution times for our protocol using the best benchmarks known to us (as discussed above), i.e. HE encryption: 8 ms, HE decryption: 17 ms.

We have also implemented the OPE scheme proposed in [5]. Our implementation was independent of the only two other implementations of such scheme in the literature known to us, which are parts of the encrypted database systems CryptDB [46] and Monomi [57]. Using typical features for implicit authentication, a maximum plaintext length around 100 bits seems to be sufficient for our protocol Π. Execution times required for encryption and decryption in our implementation are at most 56 ms for plaintext lengths between 100 and 1000 bits[5]. CryptDB authors report initial encryption time of 25 ms for 32-bit plaintexts [46], and were able to optimise the encryption to bring the encryption time down to 4 ms [47]. We use the 56-ms benchmark to calculate concrete execution times for a simple implementation of our protocol. However, as the CryptDB implementation demonstrates, execution times can be as much as around 10 times lower. Hence, we also provide concrete execution times based on the optimised benchmarks above, i.e. OPE encryption / decryption: 4 ms. This optimised benchmark assumes that features may be expressed in 32 bits, which is a reasonable assumption for the majority of features proposed in the literature for implicit authentication. In fact all the candidate features listed in Section 3.2 can be expressed in 32 bits. GPS coordinates are usually expressed in the NMEA 0183 standard format [1] which for a one meter precision would consist of at most 9 digits: at most 3 for degrees, 2 for minutes, and at most 4 for decimal minutes. 9 digits may be expressed in 30 bits. An additional bit is needed to indicate either N/S or E/W. Hence latitude and longitude can each be expressed in 32 bits. Other discussed candidates such as power consumption in percentage, WiFi session duration in minutes, battery level in percentage, and interaction time in minutes can be immediately seen to be expressible in less than 32 bits.

Using the above sets of benchmarks in two categories: simple and optimised benchmarks, we can estimate concrete times for our protocol on the device side. Table 1 summarises the computational complexity of the profile initialisation, authentication, and profile update phases of protocol Π for one feature. On the device side, initialisation phase complexity is reported for each period in which the device reports a pair of ciphertexts, whereas on the carrier-side, the computation of the initial AAD is assumed to take place at once at the end of the phase, hence the reported complexity is for the whole initialisation phase. On the carrier side, the computations are limited to ciphertext-space homomorphic additions and scalar multiplications which are in the order of hundreds of time faster than HE and OPE encryption. Besides, the carrier side naturally has much more computational power than the device side. Hence, despite the multiplicative factor ℓ_i (typically in the order of 100) in the complexity of the profile initialisation phase on the carrier side, the concrete execution time for all phases on the carrier side ends up being negligible compared to those on the device side.

Finally, note that the authentication phase for each feature takes *less than 300 ms* on the device side and negligible time on the carrier side, even with a non-optimised implementation. The concrete system example given in Section 3.2 involves at most five features and hence the total system authentication time will be at most five times the above figure. Considering the whole process is executed implicitly as a background process, the overhead of introducing privacy is not significant.

In terms of communication complexity, for each feature the device needs to send one HE ciphertext and 3 OPE

[5]OPE complexity depends on ciphertext length as well as plaintext length. In our implementations, we have considered combinations of plaintext sizes of 100 and 1000 bits and ciphertext sizes of 10, 100, and 1000 kilobits.

Phase	Complexity	Concrete Times (simple)	Concrete Times (optimised)
Device-side:			
Init.	$t_{HE} + t_{OPE}$	82 ms	12 ms
Auth.	$t_{HE} + 2t_{HD} + 3t_{OPE}$	264 ms	54 ms
Carrier-side:			
Init.	$\ell_i t_{HA} + t_{HM}$	negl.	negl.
Auth.	$2t_{HA}$	negl.	negl.
Upd.	$4t_{HA} + t_{HM}$	negl.	negl.

Table 1: Computation complexity of protocol Π based on one feature assuming $\ell_i = 100$. Legend: Init: profile initialisation phase, Auth: authentication phase, Upd: profile update phase, t_{HE}, t_{HD}: HE encryption and decryption times, t_{OPE}: OPE encryption time, t_{HA}, t_{HM}: HE ciphertext-space addition and scalar multiplication times, negl: negligible.

ciphertexts in each round of authentication, and the carrier needs to send 2 HE ciphertexts. Each HE ciphertext is 1 kb (kilobits) and each OPE ciphertext may be implemented as 10 kb for typical plaintext sizes in our scheme. This means that the device needs to send less than 4 kB (kilo Bytes) and receive around 0.25 kB in each round of communication.

Security: We discuss the security of our protocol considering semi-honest devices and carriers in Appendix B. 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 Appendix B.1, we prove the following theorem guaranteeing the privacy of our protocol:

Theorem 1 *Our protocol* Π *is provably secure against semi-honest devices and semi-honest 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 $E^{HE}_{pk}(b^{\mathrm{L}}_i(t))$ and $E^{HE}_{pk}(b^{\mathrm{H}}_i(t))$ to the device and the device is expected to reply back order-preserving ciphertexts of the same plaintexts, i.e. $E^{OPE}_{k_2}(b^{\mathrm{L}}_i(t))$ and $E^{OPE}_{k_2}(b^{\mathrm{H}}_i(t))$. These order-preserving ciphertexts are subsequently used to compare the values of $b^{\mathrm{L}}_i(t)$ and $b^{\mathrm{H}}_i(t)$ in the order-preserving ciphertext space with the feature values and find out how many feature values lie between $b^{\mathrm{L}}_i(t)$ and $b^{\mathrm{H}}_i(t)$. However, a malicious device cannot be trusted to return correctly formatted order-preserving ciphertexts.

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[6], 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 $E^{HE}_{pk}(v_i(t))$ is well-formed. To enforce this requirement, we require that the device sends a proof of knowledge of the corresponding plaintext $v_i(t)$ along with the ciphertext $E^{HE}_{pk}(v_i(t))$. Efficient proofs of knowledge of plaintext exist for most public key encryption schemes. For Paillier encryption, a concrete and efficient interactive proof protocol can be found in [4]. The protocol can be made non-interactive using the well-known Fiat-Shamir heuristic, by replacing the random challenge generated by the verifier with the hash of the protocol parameters concatenated with the message sent by the prover in the first round. We denote the resulting proof of knowledge of the plaintext $v_i(t)$ by $PoK\{v_i(t)\}$.

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 $b^{\mathrm{L}}_i(t)$ and $b^{\mathrm{H}}_i(t)$ with the feature values, only using the homomorphic ciphertexts. Assume that the carrier wishes to compare $b^{\mathrm{L}}_i(t)$ with $v_i(t_j)$. The carrier has homomorphic encryptions of both, i.e. $E^{HE}_{pk}(b^{\mathrm{L}}_i(t))$ with $E^{HE}_{pk}(v_i(t_j))$, and hence can calculate $E^{HE}_{pk}(b^{\mathrm{L}}_i(t) - v_i(t_j))$. The carrier is hence interested in knowing whether $b^{\mathrm{L}}_i(t) - v_i(t_j)$ is positive, negative, or zero. In the following, we show how the carrier is able to interact with the device and determine whether the above value is positive, negative, or zero, without the device being able to cheat or to have a noticeable chance of finding out some information about the value in question.

In the following, we propose a modified version of the protocol secure against malicious devices. We call this modified version $\Pi^\star$. Let $HE = (KeyGen^{HE}, E^{HE}, D^{HE})$ be a homomorphic encryption scheme, such as Paillier cryptosystem. The protocol $\Pi^\star$ consists of four phases: system setup, (user) profile initialisation, authentication, and profile update. The profile initialisation phase is exactly the same as that of the protocol Π described in Section 4.1, and thus is not repeated here. System setup is carried out once for each device, but afterwards the authentication and profile

[6]`www.arm.com/products/processors/technologies/trustzone`

update phases are carried out once per each authentication round. Authentication rounds are carried out periodically and continuously. The protocol works as follows:

Phase 1. System Setup: This phase is performed once for each device. $KeyGen^{HE}$ is run by the device to generate the HE key pair (pk, sk). The public key pk is communicated to the carrier. The private key sk is kept by the device.

Phase 2. Profile Initialisation: This phase is performed only once for each device to record the initial ℓ_i feature readings and compute an initial AAD for each feature. During this phase the device is assumed to be honest. This phase is similar to the user initialisation phase in protocol Π, but here there are no OPE ciphertexts involved. During this phase, the device periodically sends HE encrypted feature readings $e_i(t) = E_{pk}^{HE}(v_i(t))$ to the carrier. The device also keeps a record of $v_i(t)$ values and along with each HE ciphertext, it sends the order information of the value with respect to the previous values to the carrier. The communications end after ℓ_i feature readings. At the end of this phase, the carrier has ℓ_i ciphertexts for the i-th feature: $\{\ e_i(t_j)\ \}_{j=1}^{\ell_i}$ and the ordering information about the corresponding plaintexts. Since the carrier knows the ordering of plaintexts, it is able to find the encryption of the median of the feature readings $E_{pk}^{HE}(\mathrm{Med}(V_i))$, where $\mathrm{Med}(V_i)$ denotes the median of $\{\ v_i(t_j)\ \}_{j=1}^{\ell_i}$. The carrier finds the indexes of the top and bottom half of plaintexts with respect to the median. Let us denote the set of top half indexes by T_i and the set of bottom half indexes by B_i. The carrier uses the homomorphic property of HE to compute the encryption of AAD based on Equation 1 as follows:

$$E_{pk}^{HE}(\mathrm{AAD}(V_i)) = \ell_i^{-1} \cdot \Big(\sum_{j \in T_i} e_i(t_j) - \sum_{j \in B_i} e_i(t_j) \Big).$$

The device deletes the record of $v_i(t)$ values it has been keeping at the end of this phase. The setup and initialisation of the system are complete and from now on, the system will enter the mode in which the device is not trusted any more. In this mode, a continuous and periodic succession of authentication and profile update phases will be carried out.

Phase 3. Authentication: The device and the carrier enter the authentication phase with the carrier holding a profile of the device user including ℓ_i HE ciphertexts for the i-th feature: $\{\ e_i(t_j) = E_{pk}^{HE}(v_i(t_j))\ \}_{j=1}^{\ell_i}$ and the HE encryption of the AAD of the features $E_{pk}^{HE}(\mathrm{AAD}(V_i))$. The device reports to the carrier the HE encryption of a new reading $e_i(t) = E_{pk}^{HE}(v_i(t))$. The device also sends a proof of knowledge of the plaintext $PoK\{v_i(t)\}$ to show that the ciphertext is well-formed. The carrier verifies the proof of knowledge and if the verification fails, it deems authentication failed. Otherwise, the carrier calculates the following using the homomorphic property:

$$\begin{aligned} E_{pk}^{HE}(b_i^{\mathrm{L}}(t)) &\leftarrow E_{pk}^{HE}(v_i(t)) - E_{pk}^{HE}(\mathrm{AAD}(V_i)) \\ E_{pk}^{HE}(b_i^{\mathrm{H}}(t)) &\leftarrow E_{pk}^{HE}(v_i(t)) + E_{pk}^{HE}(\mathrm{AAD}(V_i)) \end{aligned}$$

Now the carrier needs to find out how $b_i^{\mathrm{L}}(t)$ and $b_i^{\mathrm{H}}(t)$ compare to $\{\ v_i(t_j)\ \}_{j=1}^{\ell_i}$ to be able to count the number of $e_i(t_j)$ values that fall between $b_i^{\mathrm{L}}(t)$ and $b_i^{\mathrm{H}}(t)$ for the purpose of authentication. The carrier also needs to find out how the new reported reading $v_i(t)$ compares to the previous ones $\{\ v_i(t_j)\ \}_{j=1}^{\ell_i}$, so that if the authentication succeeds, it has the ordering information necessary to update the profile accordingly in the profile update phase. Let us define for all i and j:

$$\begin{aligned} \delta_{ij}^{\mathrm{L}} &= b_i^{\mathrm{L}}(t) - v_i(t_j), \\ \delta_{ij} &= v_i(t) - v_i(t_j), \quad \text{and} \\ \delta_{ij}^{\mathrm{H}} &= b_i^{\mathrm{H}}(t) - v_i(t_j)\ . \end{aligned}$$

To compare any of the above values, i.e., $b_i^{\mathrm{L}}(t)$, $v^i(t)$ and $b_i^{\mathrm{H}}(t)$, with $v_i(t_j)$, the carrier needs to find out if the corresponding differences, i.e., δ_{ij}^{L}, δ_{ij}, and δ_{ij}^{H}, as defined above, are each negative, zero, or positive. To achieve this, the carrier first calculates for all $j \in [1, \ell_i]$ the ciphertexts $E_{pk}^{HE}(\delta_{ij}^{\mathrm{L}})$, $E_{pk}^{HE}(\delta_{ij})$, and $E_{pk}^{HE}(\delta_{ij}^{\mathrm{H}})$ using the homomorphic property of the encryption scheme from $E_{pk}^{HE}(v_i(t_j))$, $E_{pk}^{HE}(v_i(t))$, $E_{pk}^{HE}(b_i^{\mathrm{L}}(t))$, and $E_{pk}^{HE}(b_i^{\mathrm{H}}(t))$. Then the carrier chooses ℓ_i random bits, and for each $j \in [1, \ell_i]$ based on the j-th bit either leaves the calculated HE ciphertext triplets as is, or calculates the ciphertext triplets for $-\delta_{ij}^{\mathrm{L}}$, $-\delta_{ij}$, and $-\delta_{ij}^{\mathrm{H}}$ through ciphertext homomorphic scalar multiplication by -1. Let us denote these ciphertexts by $E_{pk}^{HE}(\pm\delta_{ij}^{\mathrm{L}})$, $E_{pk}^{HE}(\pm\delta_{ij})$, and $E_{pk}^{HE}(\pm\delta_{ij}^{\mathrm{H}})$. This makes sure that the these differences are distributed independently of the value of the current reading in terms of being positive or negative. That is, on any $v_i(t)$ a similar number of the differences will be positive or negative.

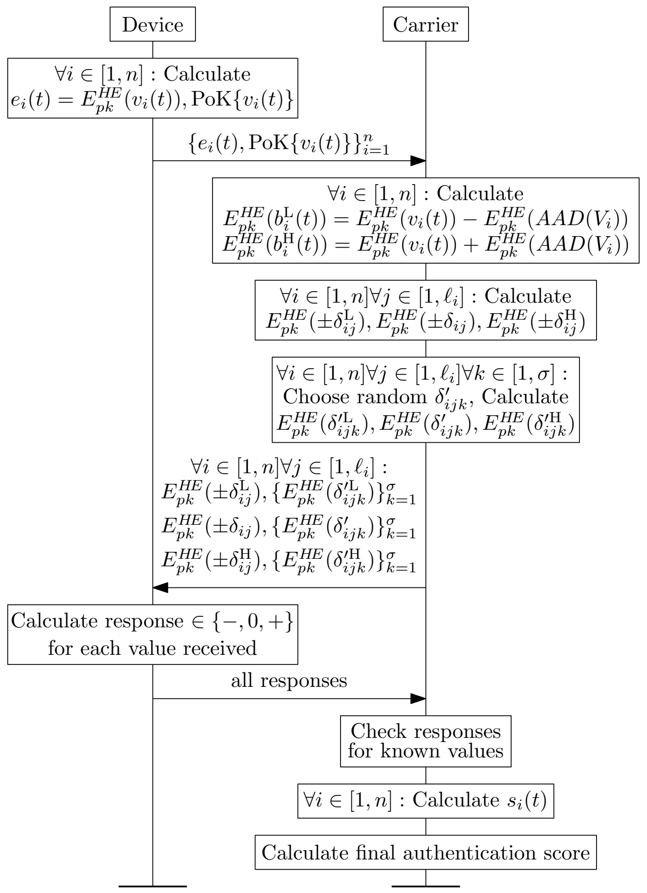
Assume the i-th feature values belong to the interval $[\min_i, \max_i]$ with a range $d_i = \max_i - \min_i$. This means $\delta_{ij} \in [-d_i, d_i]$. The carrier chooses $\sigma\ell_i$ random values $\{\{\delta'_{ijk}\}_{j=1}^{\ell_i}\}_{k=1}^{\sigma}$ from the interval $[-d_i, d_i]$, where σ is a security parameter. The values $\{\delta'_{ijk}\}_{k=1}^{\sigma}$ serve as values among which δ_{ij} will be hidden. Also note that $\delta_{ij}^{\mathrm{L}} = \delta_{ij} - \mathrm{AAD}(V_i)$ and $\delta_{ij}^{\mathrm{H}} = \delta_{ij} + \mathrm{AAD}(V_i)$. Let us define analogously $\delta_{ijk}'^{\mathrm{L}} = \delta'_{ijk} - \mathrm{AAD}(V_i)$ and $\delta_{ijk}'^{\mathrm{H}} = \delta'_{ijk} + \mathrm{AAD}(V_i)$. The carrier now calculates the corresponding ciphertexts for the "fake" difference values as follows: for all j and k it calculates $E_{pk}^{HE}(\delta'_{ijk})$, and then $E_{pk}^{HE}(\delta_{ijk}'^{\mathrm{L}})$ and $E_{pk}^{HE}(\delta_{ijk}'^{\mathrm{H}})$.

The carrier then puts together the following set of values for all $j \in [1, \ell_i]$ and all $k \in [1, \sigma]$: $E_{pk}^{HE}(\delta_{ij}^{\mathrm{L}})$, $E_{pk}^{HE}(\delta_{ij})$, $E_{pk}^{HE}(\delta_{ij}^{\mathrm{H}})$, $E_{pk}^{HE}(\delta'_{ijk})$, $E_{pk}^{HE}(\delta_{ijk}'^{\mathrm{L}})$), and $E_{pk}^{HE}(\delta_{ijk}'^{\mathrm{H}})$. The carrier shuffles these values and sends them to the device. The device decrypts the ciphertexts and replies to the carrier indicating whether each ciphertext corresponds to a positive, zero, or negative plaintext. The device is able to compute $\mathrm{AAD}(V_i)$ and also distinguish the three sets of values: the differences δ_{ij} and δ'_{ijk}, the differences minus AAD, and the differences plus AAD. However, among the differences δ_{ij} and δ'_{ijk} the device cannot distinguish between "real" and "fake" values. The carrier on the other hand, knows what the response should be for all fake differences δ'_{ijk}. Also if δ_{ij} is positive, then the carrier knows that δ^{H}_{ij} should be also positive, and if δ_{ij} is negative, then the carrier knows that δ^{L}_{ij} should be also negative. Hence upon receiving the responses the carrier checks if these responses are correct and if not the authentication is deemed failed. The idea here is that since all the $\sigma + 1$ differences (real and fake altogether) look indistinguishable to the device, a malicious device has at most $\frac{1}{\sigma+1}$ chance of cheating and not getting caught. σ is a security parameter of the protocol and controls a trade-off between complexity and security. The larger σ 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.

If the responses pass all the checks, from the responses for the real differences, the carrier is able to find out how each of $b^{\mathrm{L}}_i(t)$, $v_i(t)$, and $b^{\mathrm{H}}_i(t)$ compare to $\{\ v_i(t_j)\ \}_{j=1}^{\ell_i}$. The carrier computes the individual score $s_i(t)$ as the number of $v_i(t_j)$ that are between $b^{\mathrm{L}}_i(t)$ and $b^{\mathrm{H}}_i(t)$. The final authentication decision is then made by the carrier based on its authentication policy, e.g. the weighted sum method described earlier in Section 3.1. 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. If either implicit or explicit authentication is successful, the carrier enters the profile update phase. Figure 4 shows the interaction diagram of the authentication phase of the protocol.

Phase 4. Profile Update: The carrier enters this phase after a successful implicit or explicit authentication. The carrier updates the recorded features and the AAD in this phase. The calculations in this phase are the same as those of the profile update phase in protocol Π. At the end of this phase, the carrier holds a set of updated feature ciphertexts and an updated AAD ciphertext. The carrier will enter the authentication phase afterwards and wait for a new feature reading to be reported from the device.

Complexity: We discuss the computation complexity of the profile initialisation, authentication, and profile update phases of our protocol $\Pi^\star$ in the following. We also calculate concrete running times for the protocol. Like before, we analyse the computation complexity of the protocol for one feature. To calculate approximate execution times for multiple features, the figures may be multiplied by the number of features.



Device
Carrier
$\forall i \in [1,n]$: Calculate $e_i(t) = E^{HE}_{pk}(v_i(t)), \mathrm{PoK}\{v_i(t)\}$
$\{e_i(t), \mathrm{PoK}\{v_i(t)\}\}_{i=1}^n$
$\forall i \in [1,n]$: Calculate $E^{HE}_{pk}(b^{\mathrm{L}}_i(t)) = E^{HE}_{pk}(v_i(t)) - E^{HE}_{pk}(AAD(V_i))$ $E^{HE}_{pk}(b^{\mathrm{H}}_i(t)) = E^{HE}_{pk}(v_i(t)) + E^{HE}_{pk}(AAD(V_i))$
$\forall i \in [1,n] \forall j \in [1,\ell_i]$: Calculate $E^{HE}_{pk}(\pm\delta^{\mathrm{L}}_{ij}), E^{HE}_{pk}(\pm\delta_{ij}), E^{HE}_{pk}(\pm\delta^{\mathrm{H}}_{ij})$
$\forall i \in [1,n] \forall j \in [1,\ell_i] \forall k \in [1,\sigma]$: Choose random δ'_{ijk}, Calculate $E^{HE}_{pk}(\delta'^{\mathrm{L}}_{ijk}), E^{HE}_{pk}(\delta'_{ijk}), E^{HE}_{pk}(\delta'^{\mathrm{H}}_{ijk})$
$\forall i \in [1,n] \forall j \in [1,\ell_i]$: $E^{HE}_{pk}(\pm\delta^{\mathrm{L}}_{ij}), \{E^{HE}_{pk}(\delta'^{\mathrm{L}}_{ijk})\}_{k=1}^{\sigma}$ $E^{HE}_{pk}(\pm\delta_{ij}), \{E^{HE}_{pk}(\delta'_{ijk})\}_{k=1}^{\sigma}$ $E^{HE}_{pk}(\pm\delta^{\mathrm{H}}_{ij}), \{E^{HE}_{pk}(\delta'^{\mathrm{H}}_{ijk})\}_{k=1}^{\sigma}$
Calculate response $\in \{-, 0, +\}$ for each value received
all responses
Check responses for known values
$\forall i \in [1,n]$: Calculate $s_i(t)$
Calculate final authentication score

Figure 4: The authentication phase of our protocol $\Pi^\star$

The profile initialisation and update phases are similar to those of protocol Π with the exception that OPE ciphertexts are no more involved.

The authentication phase on the other hand differs substantially from that of protocol Π. In the authentication phase, the protocol requires 1 homomorphic encryption, 1 proof of knowledge generation, and $(\sigma + 1)\ell_i$ homomorphic decryptions on the device side. Given that the proof of knowledge generation takes only a couple of multiplications, the computation complexity here is dominated by $(\sigma + 1)\ell_i$ homomorphic decryptions. On the carrier side, the following computations are required: 1 proof of knowledge verification (roughly as complex as 1 multiplication), 2 homomorphic ciphertext additions to calculate $E^{HE}_{pk}(b^{\mathrm{L}}_i(t))$ and $E^{HE}_{pk}(b^{\mathrm{H}}_i(t))$ (roughly as expensive as a multiplication each), then $3\ell_i$ homomorphic ciphertext additions to calculate $E^{HE}_{pk}(\delta^{\mathrm{L}}_{ij})$, $E^{HE}_{pk}(\delta_{ij})$, and $E^{HE}_{pk}(\delta^{\mathrm{H}}_{ij})$, then an expected $\frac{1}{2}\ell_i$ homomorphic ciphertext scalar multiplications to calculate $E^{HE}_{pk}(\pm\delta^{\mathrm{L}}_{ij})$, $E^{HE}_{pk}(\pm\delta_{ij})$, and $E^{HE}_{pk}(\pm\delta^{\mathrm{H}}_{ij})$, then $\sigma\ell_i$ homomorphic encryptions to calculate $E^{HE}_{pk}(\delta'_{ijk})$, and finally $2\sigma\ell_i$ homomorphic ciphertext additions to calculate $E^{HE}_{pk}(\delta'^{\mathrm{L}}_{ijk}))$ and $E^{HE}_{pk}(\delta'^{\mathrm{H}}_{ijk})$. This means on the carrier side the total computation cost is dominated by $\sigma\ell_i$ homomorphic encryption operations.

Choosing a small σ means that a malicious device is caught at the time of protocol execution with lower probability, however, the device does not gain meaningful advantage by cheating and will not have a higher chance of succeeding in authentication. Hence, even a small σ provides a reasonable level of protection against malicious devices. Consequently, we consider σ to be a small multiplicative factor and will be able to state that the complexity of the modified protocol is approximately proportional to ℓ_i. In other words, the complexity grows linearly with the size of the user profile.

Note that finding how each of the ciphertexts $E^{HE}_{pk}(b^{\text{L}}_i(t))$, $E^{HE}_{pk}(v_i(t))$, and $E^{HE}_{pk}(b^{\text{H}}_i(t))$ compare with the recorded features can be carried out in $\log \ell_i$ rounds (instead of at once) through a binary search. That is, since the carrier knows the ordering of the recorded profile features, in each round the carrier can ask the device to help with comparing the above ciphertexts with one recorded feature value, and based on the answer to each round decide which recorded feature value to use for comparison in the next round. This is a trade-off between the round complexity and the communication complexity. Carrying out the comparison in this way requires $\log \ell_i$ rounds of communication (instead of one), $\sigma \log \ell_i$ homomorphic encryption operations on the server side, and $\sigma \log \ell_i$ homomorphic decryption operations on the client side. Thus this trade-off brings the communication complexity down to a logarithmic function of the size of the user profile. We consider this a reasonable price to be paid for protection against malicious devices.

To give concrete examples, consider $\sigma = 9$ (which means a cheating device is caught immediately with probability $\frac{1}{10}$ each time it deviates from the protocol) and a typical profile size of $\ell_i = 100$.

Table 2 summarises the computational complexity of the profile initialisation, authentication, and profile update phases of protocol $\Pi^\star$ for one feature, using the same benchmarks as in the previous section. Similar to before, on the device side, initialisation phase complexity is reported for each period, whereas on the carrier-side, the reported complexity is for the whole initialisation phase. On the carrier side, the computations in the profile initialisation and update phases are limited to ciphertext-space homomorphic additions and scalar multiplications which end up being negligible compared to the other computation times. The authentication phase however requires $\sigma \log \ell_i$ homomorphic encryptions on the carrier side. To merely calculate a nominal concrete execution time for the carrier-side, we assume that the carrier has 10 times the processing power of the device. This assumption gives us the figures for concrete execution times for the authentication phase on the carrier side reported in Table 2. Of course, the concrete figures in this case are to be treated as merely an indication of the efficiency of the protocol.

Phase	Complexity	Concrete Times	
		(simple)	(optimised)
Device-side:			
Init.	t_{HE}	26 ms	8 ms
Auth.	$(\sigma+1)\log \ell_i t_{HD}$	2326 ms	1130 ms
Carrier-side:			
Init.	$\ell_i t_{HA} + t_{HM}$	negl.	negl.
Auth.	$\sigma \log \ell_i t_{HE}$	156 ms	48 ms
Upd.	$4t_{HA} + t_{HM}$	negl.	negl.

Table 2: Computation complexity of protocol $\Pi^\star$ based on one feature assuming $\sigma = 9$, $\ell_i = 100$, and that the carrier side has 10 times the computation power of the device side. Legend: Init: profile initialisation phase, Auth: authentication phase, Upd: profile update phase, t_{HE}, t_{HD}: HE encryption and decryption times, t_{HA}, t_{HM}: HE ciphertext-space addition and scalar multiplication times, negl: negligible.

Finally, considering the execution times on both sides, note that an authentication failure for one feature is discovered *in around 2.5 seconds* after the first feature reading is reported by the device, even with a non-optimised implementation. The concrete system example given in Section 3.2 involves at most five features and hence the total system authentication time will be at most five times the above figure. We stress again that implicit authentication is an ongoing background process and does not need to be real-time.

In terms of communication complexity, for each feature the device needs to send one HE ciphertext, one proof of knowledge, and $3(\sigma+1)\log \ell_i$ bits in each round of authentication, and the carrier needs to send $3(\sigma+1)\log \ell_i$ HE ciphertexts. Each HE ciphertext is 1 kb (kilobits) and each proof of knowledge is 2 kb for typical parameters in our scheme. This means that the device needs to send less than 0.5 kB (kilo Bytes) and receive around 25 kB in each round of communication.

Security: We discuss the security of our protocol considering malicious devices in Appendix C. 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 Appendix C.1, we prove the following theorem guaranteeing the privacy of our protocol:

Theorem 2 *Our protocol $\Pi^\star$ is provably secure against maliciously-controlled devices (with probability at least $\frac{\sigma}{\sigma+1}$), 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. Our benchmark implementations and other optimised implementations of the primitives used in our protocols give us concrete estimations of execution times for our protocols. We provided such concrete times and argued that our protocols are sufficiently efficient in practice.

Acknowledgements

The authors would like to thank the anonymous reviewers of Elsevier's Computers and Security as well as those of IFIP SEC 2014 for their constructive comments that improved this article considerably.

References

[1] The NMEA 0183 Standard. The National Marine Electronics Association. `http://www.nmea.org`.

[2] F. Aloul, S. Zahidi, and W. El-Hajj. Two Factor Authentication Using Mobile Phones. In *Computer Systems and Applications (AICCSA 2009), IEEE/ACS Int'l Conf. on*, pages 641–644. IEEE, 2009.

[3] A. Basu, H. Kikuchi, and J. Vaidya. Privacy-Preserving Weighted Slope One predictor for Item-based Collaborative Filtering. In *Proceedings of the int'l workshop on Trust and Privacy in Distributed Information Sharing (IFIP TP-DIS 2011)*, 2011.

[4] O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard. Practical Multi-Candidate Election System. In *Proc. 20th ACM symposium on Principles of Distributed Computing*, pages 274–283. ACM, 2001.

[5] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-Preserving Symmetric Encryption. In *Advances in Cryptology - EUROCRYPT 2009*, pages 224–241. Springer, 2009.

[6] A. Boldyreva, N. Chenette, and A. O'Neill. Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions. In *Advances in Cryptology - CRYPTO 2011*, pages 578–595. Springer, 2011.

[7] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman. Semantically Secure Order-Revealing Encryption: Multi-Input Functional Encryption Without Obfuscation. In *Proceedings of EuroCrypt 2015 (to appear)*, 2015. Preprint available at `http://eprint.iacr.org/2014/834`.

[8] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure Remote Authentication Using Biometric Data. In *Advances in Cryptology - EUROCRYPT 2005*, pages 147–163. Springer, 2005.

[9] S. Čapkun, M. Čagalj, and M. Srivastava. Secure Localization with Hidden and Mobile Base Stations. In *Int'l Conf. on Computer Communication (INFOCOM 2006)*, 2006.

[10] S. Čapkun and J.-P. Hubaux. Secure Positioning of Wireless Devices with Application to Sensor Networks. In *INFOCOM 2005: 24th Annual Joint Conf. of the IEEE Computer and Communications Societies*, volume 3, pages 1917–1928. IEEE, 2005.

[11] K.-H. Chang, J. Hightower, and B. Kveton. Inferring Identity Using Accelerometers in Television Remote Controls. In *Pervasive Computing*, pages 151–167. Springer, 2009.

[12] J. T. Chiang, J. J. Haas, and Y.-C. Hu. Secure and Precise Location Verification Using Distance Bounding and Simultaneous Multilateration. In *2nd ACM conference on Wireless Network Security*, pages 181–192. ACM, 2009.

[13] R. Chow, M. Jakobsson, R. Masuoka, J. Molina, Y. Niu, E. Shi, and Z. Song. Authentication in the Clouds: A Framework and Its Application to Mobile Users. In *Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*, CCSW '10, pages 1–6, New York, NY, USA, 2010. ACM.

[14] N. Clarke and S. Furnell. Authenticating Mobile Phone Users Using Keystroke Analysis. *International Journal of Information Security*, 6(1):1–14, 2007.

[15] I. Damgård and M. Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *Public Key Cryptography*, pages 119–136. Springer, 2001.

[16] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann. Touch Me Once and I Know It's You!: Implicit Authentication Based on Touch Screen Patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 987–996, New York, NY, USA, 2012. ACM.

[17] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In *Advances in cryptology - Eurocrypt 2004*, pages 523–540. Springer, 2004.

[18] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. Diversity in Smartphone Usage. In *Proceedings of the 8th Int'l Conf. on Mobile Systems, Applications, and Services*, MobiSys '10, pages 179–194. ACM, 2010.

[19] T. Feng, X. Zhao, B. Carbunar, and W. Shi. Continuous Mobile Authentication Using Virtual Key Typing Biometrics. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*, pages 1547–1552, July 2013.

[20] J. Frank, S. Mannor, and D. Precup. Activity and Gait Recognition with Time-Delay Embeddings, 2010.

[21] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song. Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication. *Information Forensics and Security, IEEE Transactions on*, 8(1):136–148, Jan 2013.

[22] S. Furnell, N. Clarke, and S. Karatzouni. Beyond the PIN: Enhancing User Authentication for Mobile Devices. *Computer Fraud & Security*, 2008(8):12–17, 2008.

[23] D. Gafurov, K. Helkala, and T. Søndrol. Biometric Gait Authentication Using Accelerometer Sensor. *Journal of Computers*, 1(7):51–59, 2006.

[24] C. Gentry. *A Fully Homomorphic Encryption Scheme.* PhD thesis, Stanford University, 2009.

[25] C. Gentry and S. Halevi. Implementing Gentrys Fully-Homomorphic Encryption Scheme. In *Advances in Cryptology - EUROCRYPT 2011*, pages 129–148. Springer, 2011.

[26] O. Goldreich, S. Micali, and A. Wigderson. How to Play Any Mental Game - A Completeness Theorem for Protocols with Honest Majority. In *Proc. 19th ACM symposium on Theory of Computing*, pages 218–229. ACM, 1987.

[27] E. Haubert, J. Tucek, L. Brumbaugh, and W. Yurcik. Tamper-Resistant Storage Techniques for Multimedia Systems. In *Electronic Imaging 2005*, pages 30–40. International Society for Optics and Photonics, 2005.

[28] S.-s. Hwang, S. Cho, and S. Park. Keystroke dynamics-based authentication for mobile devices. *Computers & Security*, 28(1–2):85–93, 2009.

[29] T. Jakobsen, M. Makkes, and J. Nielsen. Efficient Implementation of the Orlandi Protocol. In J. Zhou and M. Yung, editors, *Applied Cryptography and Network Security*, volume 6123 of *Lecture Notes in Computer Science*, pages 255–272. Springer Berlin Heidelberg, 2010.

[30] M. Jakobsson, E. Shi, P. Golle, and R. Chow. Implicit Authentication for Mobile Devices. In *Proc. of the 4th USENIX conf. on Hot Topics in Security*. USENIX Association, 2009.

[31] V. Kachitvichyanukul and B. Schmeiser. Computer Generation of Hypergeometric Random Variates. *Journal of Statistical Computation and Simulation*, 22(2):127–145, 1985.

[32] A. Kalamandeen, A. Scannell, E. de Lara, A. Sheth, and A. LaMarca. Ensemble: Cooperative Proximity-based Authentication. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 331–344, New York, NY, USA, 2010. ACM.

[33] A. Kale, A. Rajagopalan, N. Cuntoor, and V. Krüger. Gait-Based Recognition of Humans Using Continuous HMMs. In *Proc. 5th IEEE Int'l Conf. on Automatic Face & Gesture Recognition*, pages 336–341. IEEE, 2002.

[34] J.-M. Kang, S.-S. Seo, and J. W.-K. Hong. Usage Pattern Analysis of Smartphones. In *13th Asia-Pacific Network Operations and Management Symposium (APNOMS '11)*, pages 1–8. IEEE, 2011.

[35] J. Krumm. Inference Attacks on Location Tracks. In *Pervasive Computing*, pages 127–143. Springer, 2007.

[36] J. Leggett, G. Williams, M. Usnick, and M. Longnecker. Dynamic Identity Verification via Keystroke Characteristics. *International Journal of Man-Machine Studies*, 35(6):859–870, 1991.

[37] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The Jigsaw Continuous Sensing Engine for Mobile Phone Applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 71–84. ACM, 2010.

[38] E. Maiorana, P. Campisi, N. González-Carballo, and A. Neri. Keystroke Dynamics Authentication for Mobile Phones. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 21–26, New York, NY, USA, 2011. ACM.

[39] S. Möller, C. Perlov, W. Jackson, C. Taussig, and S. R. Forrest. A Polymer/Semiconductor Write-Once Read-Many-Times Memory. *Nature*, 426(6963):166–169, 2003.

[40] F. Monrose and A. Rubin. Authentication via Keystroke Dynamics. In *Proceedings of the 4th ACM conference on Computer and Communications Security*, pages 48–56. ACM, 1997.

[41] M. Nauman, T. Ali, and A. Rauf. Using trusted computing for privacy preserving keystroke-based authentication in smartphones. *Telecommunication Systems*, 52(4):2149–2161, 2013.

[42] M. Nisenson, I. Yariv, R. El-Yaniv, and R. Meir. Towards Behaviometric Security Systems: Learning to Identify a Typist. In *Knowledge Discovery in Databases: PKDD 2003*, pages 363–374. Springer, 2003.

[43] L. O'Gorman. Comparing Passwords, Tokens, and Biometrics for User Authentication. *Proceedings of the IEEE*, 91(12):2021–2040, 2003.

[44] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in cryptology - EUROCRYPT99*, pages 223–238. Springer, 1999.

[45] B. Parno, J. McCune, and A. Perrig. Bootstrapping Trust in Commodity Computers. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 414–429, May 2010.

[46] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. CryptDB: Protecting Confidentiality with Encrypted Query Processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 85–100, New York, NY, USA, 2011. ACM.

[47] R. A. Popa, N. Zeldovich, and H. Balakrishnan. CryptDB: A Practical Encrypted Relational DBMS. Technical Report MIT-CSAIL-TR-2011-005, Computer Science and Artificial Intelligence Lab (CSAIL), Massachusetts Institute of Technology, 2011. Available at `http://hdl.handle.net/1721.1/60876`.

[48] O. Riva, C. Qin, K. Strauss, and D. Lymberopoulos. Progressive Authentication: Deciding When to Authenticate on Mobile Phones. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, pages 301–316, Bellevue, WA, 2012. USENIX.

[49] N. A. Safa, R. Safavi-Naini, and S. F. Shahandashti. Privacy-Preserving Implicit Authentication. In N. Cuppens-Boulahia, F. Cuppens, S. Jajodia, A. Abou El Kalam, and T. Sans, editors, *ICT Systems Security and Privacy Protection*, volume 428 of *IFIP Advances in Information and Communication Technology*, pages 471–484. Springer Berlin Heidelberg, 2014.

[50] S. F. Shahandashti, R. Safavi-Naini, and P. Ogunbona. Private Fingerprint Matching. In *Information Security and Privacy*, pages 426–433. Springer, 2012.

[51] S. F. Shahandashti, R. Safavi-Naini, and N. A. Safa. Reconciling User Privacy and Implicit Authentication for Mobile Devices. *Computers & Security*, 2015. (`DoI: 10.1016/j.cose.2015.05.009`).

[52] E. Shi, Y. Niu, M. Jakobsson, and R. Chow. Implicit Authentication through Learning User Behavior. In M. Burmester, G. Tsudik, S. Magliveras, and I. Ilić, editors, *Information Security*, volume 6531 of *Lecture Notes in Computer Science*, pages 99–113. Springer Berlin Heidelberg, 2011.

[53] D. Singelee and B. Preneel. Location Verification Using Secure Distance Bounding Protocols. In *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pages 840–846. IEEE, 2005.

[54] A. Studer and A. Perrig. Mobile User Location-specific Encryption (MULE): Using Your Office As Your Password. In *Proceedings of the Third ACM Conference on Wireless Network Security*, WiSec '10, pages 151–162, New York, NY, USA, 2010. ACM.

[55] K. Tan, G. Yan, J. Yeo, and D. Kotz. 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*, pages 33–36. ACM, 2010.

[56] C.-S. Tsai, C.-C. Lee, and M.-S. Hwang. Password Authentication Schemes: Current Status and Key Issues. *IJ Network Security*, 3(2):101–115, 2006.

[57] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich. Processing Analytical Queries over Encrypted Data. *Proc. VLDB Endow.*, 6(5):289–300, Mar. 2013.

[58] D.-S. Wang and J.-P. Li. A New Fingerprint-Based Remote User Authentication Scheme Using Mobile Devices. In *Int'l Conf. on Apperceiving Computing and Intelligence Analysis (ICACIA 2009)*, pages 65–68. IEEE, 2009.

[59] H. Xu, Y. Zhou, and M. R. Lyu. Towards Continuous and Passive Authentication via Touch Biometrics: An Experimental Study on Smartphones. In *Symposium On Usable Privacy and Security (SOUPS 2014)*, pages 187–198, Menlo Park, CA, July 2014. USENIX Association.

[60] A. C.-C. Yao. How to Generate and Exchange Secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986.

[61] S. Zahid, M. Shahzad, S. A. Khayam, and M. Farooq. Keystroke-Based User Identification on Smart Phones. In E. Kirda, S. Jha, and D. Balzarotti, editors, *Recent Advances in Intrusion Detection*, volume 5758 of *Lecture Notes in Computer Science*, pages 224–243. Springer Berlin Heidelberg, 2009.

A Order Preserving Encryption

Consider an order-preserving (symmetric) encryption defined as $OPE = (KeyGen^{OPE}, E^{OPE}, D^{OPE})$, with key space K, plaintext space D, and ciphertext space R, in which we have $|D| \leq |R|$. For an adversary A attacking the scheme, we define its *POPF-CCA advantage* (pseudo-random order-preserving function advantage under chosen-ciphertext attack) against OPE as the difference between the probability $\Pr[k \in_R K : A^{E_k^{OPE}(\cdot), D_k^{OPE}(\cdot)} = 1]$ and the probability $\Pr[f \in_R OPF_{D \mapsto R} : A^{f(\cdot), f^{-1}(\cdot)} = 1]$, where $OPF_{D \mapsto R}$ represents the set of all order-preserving functions from D to R. We say that OPE is POPF-CCA-secure if no polynomial-time adversary has a non-negligible advantage against it.

Informally, the definition implies that OPE acts indistinguishably as a random order-preserving function, even if the adversary is given free access to encrypt and decrypt arbitrary messages of its choosing. For details of such an encryption scheme, readers are referred to [5]. The OPE scheme makes use of the implementation of hypergeometric distribution (HyG) given in [31].

B Security of Protocol Π

To formulate a private score computing protocol, we first need to formalise a score computing protocol without privacy. We define such a protocol as follows:

Definition 1 *A* ***score computing protocol*** *for feature V_i is a protocol between a device with input $Z_i = (v_i(t), t)$ and a carrier with input Y_i, where t denotes the current time, $v_i(t)$ denotes the current feature sample, and Y_i is a sample distribution of V_i with average absolute deviation* $\mathrm{AAD}(V_i)$*. The two parties also share an input which includes agreed protocol setup parameters. The protocol output for the carrier is a score $s_i(t)$ and* NULL *for the device. The score is defined as $s_i(t) = \Pr[\ b_i^{\mathrm{L}}(t) \leq V_i \leq b_i^{\mathrm{H}}(t)\ ]$ where $b_i^{\mathrm{L}}(t) = v_i(t) - \mathrm{AAD}(V_i)$ and $b_i^{\mathrm{H}}(t) = v_i(t) + \mathrm{AAD}(V_i)$.*

Let us first consider *honest-but-curious* (a.k.a. semi-honest) adversaries. An honest-but-curious party follows the protocol, but tries to infer extra information from the protocol execution. To formalise the security of SC protocols, we will use the standard simulation-based approach. The *view* of a party in a protocol execution is a tuple consisting the party's input, random selections and all the messages it receives during an execution of the protocol. This tuple is a function of the inputs of the parties and their randomness. Let $View_D^{\Pi}(Z_i, Y_i)$ (resp. $View_S^{\Pi}(Z_i, Y_i)$), denote the random variable representing the view of the device D (resp. carrier S), with device input Z_i and carrier input Y_i, and $\overset{c}{\equiv}$ denote computational indistinguishability.

Π is said to be a *perfectly private* score computing protocol, if there exists a probabilistic polynomial time algorithm Sim_D (resp. Sim_S) that can simulate the view of D (resp. S) in Π, given only the device's input Z_i (resp. carrier's input Y_i and its output s_i); that is for all Z_i and Y_i:

$$\begin{aligned} View_D^{\Pi}(Z_i, Y_i) &\overset{c}{\equiv} Sim_D(Z_i) \\ (\text{ resp. } View_S^{\Pi}(Z_i, Y_i) &\overset{c}{\equiv} Sim_S(Y_i, s_i)\ ) \end{aligned}$$

To achieve the above security level, one can design a protocol using a fully homomorphic encryption system [25], or using a general two party computation protocol. However the communication and computation cost of these approaches will be prohibitive. For example, Gentry's fully homomorphic encryption scheme takes 32 seconds on a typical processor to perform a single re-crypt operation when the modulus is 2048 bits [24, 25].

To improve efficiency, we sacrifice perfect privacy of the protocol and allow the device and carrier to learn some aggregate and order information about the profile data, respectively. We argue that although this means some leakage of information, no direct values are revealed and the leaked information does not affect privacy of the user data in any significant way. It does not increase the adversary's success chance in authentication in a significant way either.

We therefore consider the protocol private if the device only learns the average absolute deviation (AAD) of V_i stored in the profile $\mathcal{U}$ and the carrier only learns the information that can be implied from the output of an ideal random order-preserving function f on input Z_i, i.e., only the information that can be implied from $f(Z_i)$. The information implied from such a function is shown to be little other than the order of the device input with respect to the stored data. In fact, Boldyreva et al. have proven that such a function leaks neither the precise value of any input nor the precise distance between any two inputs [6].

Alternatively, one may use an *order-revealing encryption (ORE)* instead of an OPE. OREs provide a similar functionality and may be employed as a building block in our protocols with little change required. Recently, OREs have been shown to leak strictly less information than OPEs. Boneh et al. have shown that their ORE construction, although computationally more expensive, reveals no information other than the order of plaintexts [7].

We note that, although knowing the AAD or the order of a data set does leak some information, it reveals little about the actual content. For example, the sets $\{8, 1, 4, 3, 11\}$ and $\{130, 121, 127, 125, 131\}$ have the same order and the same AAD with completely different elements. Similarly two sets of GPS coordinates may have the same order and average absolute deviation but be completely different, and in fact belong to completely different places.

To formalise our notion of privacy, let us define the augmented tuple V_i^+ that besides the elements in V_i includes $v_i(t)$, i.e. for $V_i = (v_i(t_1), v_i(t_2), \ldots, v_i(t_{\ell_i}))$ we have $V_i^+ = (v_i(t_1), v_i(t_2), \ldots, v_i(t_{\ell_i}), v_i(t))$. Also let f be an ideal random order-preserving function. Let $I^f(V_i^+)$ denote the information about V_i^+ that can be implied from $f(V_i^+)$. We emphasise again that it has been proven that $I^f(V_i^+)$ includes little more than the order information of the elements of V_i^+. Hence practically one can think of $I^f(V_i^+)$ as the information on how elements of V_i^+ are ordered. We define a private score computing protocol as follows:

Definition 2 *Let D and S denote the device and carrier entities in Π, respectively. We say that Π is a* ***private score computing protocol for honest-but-curious devices (resp. carriers)****, if there exists a probabilistic polynomial time algorithm Sim_D (resp. Sim_S for any random order-preserving function f) to simulate the view of D (resp. S) in Π, given the device's input Z_i (resp. carrier's input Y_i and its output s_i) and the average absolute deviation of V_i in $\mathcal{U}$ (resp. $I^f(V_i^+)$); that is for all Z_i and Y_i:*

$$View_D^\Pi(Z_i, Y_i) \overset{c}{\equiv} Sim_D(Z_i, \mathrm{AAD}(V_i))$$

$$(\ \textit{resp. } View_S^\Pi(Z_i, Y_i) \overset{c}{\equiv} Sim_S(Y_i, s_i, I^f(V_i^+))\ ).$$

Intuitively, the above definition requires that the information revealed to the parties during the protocol execution is limited merely to the AAD of the stored data, or little other than the order information of the current sample with respect to the stored data, respectively.

B.1 Proof of Theorem 1

Proof: (Outline) In Π, the device has the input Z_i, and receives the values $Z_i - \mathrm{AAD}(V_i)$ and $Z_i + \mathrm{AAD}(V_i)$ from the carrier during the protocol execution. Therefore,

$$View_D^\Pi(Z_i, Y_i) = (\ Z_i, Z_i - \mathrm{AAD}(V_i), Z_i + \mathrm{AAD}(V_i)\ ).$$

The device has no output at the end of the protocol. Now, let us define Sim_D such that for given inputs $(Z_i, \mathrm{AAD}(V_i))$ (according to Definition 2), it outputs $(Z_i, Z_i - \mathrm{AAD}(V_i), Z_i + \mathrm{AAD}(V_i)$, where $V_i \in \mathcal{U})$. So, for all Z_i and Y_i, the distribution $Sim_D(Z_i, \mathrm{AAD}(V_i))$ and $View_D^\Pi(Z_i, Y_i)$ are indistinguishable. Hence the protocol is secure against honest-but-curious devices.

The carrier has the input Y_i and during the execution of Π it receives the following values: $E_{pk}^{HE}(Z_i)$, $E_{k_2}^{OPE}(Z_i)$, $E_{k_2}^{OPE}(b_i^{\mathrm{L}}(t))$ and $E_{k_2}^{OPE}(b_i^{\mathrm{H}}(t))$. Therefore, for its view of the protocol, we have

$$View_S^\Pi(Z_i, Y_i) =$$
$$(\ Y_i, E_{pk}^{HE}(Z_i), E_{k_2}^{OPE}(Z_i), E_{k_2}^{OPE}(b_i^{\mathrm{L}}(t)), E_{k_2}^{OPE}(b_i^{\mathrm{H}}(t))\ ),$$

where $b_i^{\mathrm{L}}(t) = Z_i - \mathrm{AAD}(V_i)$ and $b_i^{\mathrm{H}}(t) = Z_i + \mathrm{AAD}(V_i)$. The carrier has the output $s_i(t)$.

Let $Sim_S(Y_i, s_i, I^f(V_i^+))$ be defined as follows. On inputs Y_i, s_i, and $I^f(V_i^+)$, and for a given random order-preserving function f, it first selects a random $\hat{Z}_i$ such that $\hat{Z}_i$ satisfies the information that $I^f(V_i^+)$ includes about Z_i and in particular the order relations between Z_i and elements of V_i. At the same time we require that $\hat{Z}_i$ is chosen in a way that it achieves the score s_i with respect to V_i, i.e., the number of elements in V_i that lie within the distance $\mathrm{AAD}(V_i)$ of $\hat{Z}_i$ is s_i. This is possible by shifting $\hat{Z}_i$. Then Sim_S computes and outputs the following: Y_i, $E_{pk}^{HE}(\hat{Z}_i)$, $f(\hat{Z}_i)$, $f(\hat{Z}_i - \mathrm{AAD}(V_i))$, and $f(\hat{Z}_i + \mathrm{AAD}(V_i))$.

We claim that the distribution of this output is indistinguishable from the distribution of $View_S^\Pi(Z_i, Y_i)$ for all Z_i and Y_i. If not, a standard hybrid argument implies that at least one of the following is true:

(A) there exists an algorithm that distinguishes $E_{pk}^{HE}(\hat{Z}_i)$ and $E_{pk}^{HE}(Z_i)$; or

(B) there exists an algorithm that distinguishes the tuple

$$(\ f(\hat{Z}_i),\ f(\hat{Z}_i - \mathrm{AAD}(V_i)),\ f(\hat{Z}_i + \mathrm{AAD}(V_i))\ )$$

and the tuple

$$(\ E_{k_2}^{OPE}(Z_i),\ E_{k_2}^{OPE}(Z_i - \mathrm{AAD}(V_i)),$$
$$E_{k_2}^{OPE}(Z_i + \mathrm{AAD}(V_i))\ )\ .$$

The former, (A), contradicts the semantic security of the homomorphic encryption scheme HE. We prove in the following that the latter, (B), contradicts the POPF security of the order preserving encryption OPE.

Assume (B) is true. It follows that there is a distinguisher for at least one of the following pairs: $f(\hat{Z}_i)$ and $E_{k_2}^{OPE}(Z_i)$, or $f(\hat{Z}_i - \mathrm{AAD}(V_i))$ and $E_{k_2}^{OPE}(Z_i - \mathrm{AAD}(V_i))$, or $f(\hat{Z}_i + \mathrm{AAD}(V_i))$ and $E_{k_2}^{OPE}(Z_i + \mathrm{AAD}(V_i))$. We consider these possibilities next.

Assume there is a distinguisher for $f(\hat{Z}_i)$ and $E_{k_2}^{OPE}(Z_i)$. A hybrid argument implies that there must be a distinguisher for at least one of the following pairs: $f(\hat{Z}_i)$ and $f(Z_i)$, or $f(Z_i)$ and $E_{k_2}^{OPE}(Z_i)$. A distinguisher for the former pair is impossible because $\hat{Z}_i$ is chosen to conform to $I^f(V_i^+)$, i.e. the information implied from either of $f(\hat{Z}_i)$ or $f(Z_i)$ is the same. A distinguisher for the latter pair on the other hand implies that it is possible to distinguish the order-preserving encryption OPE from f, which contradicts the security of the OPE.

Now note that since $\text{AAD}(V_i)$ is a constant determined by Y_i, the three distributions Z_i, $Z_i - \text{AAD}(V_i)$, and $Z_i + \text{AAD}(V_i)$ are merely shifted versions of one another. The same is true for $\hat{Z}_i$, $\hat{Z}_i - \text{AAD}(V_i)$, and $\hat{Z}_i + \text{AAD}(V_i)$. Hence, similar arguments can be made to show that a distinguisher for any of the pairs $f(\hat{Z}_i - \text{AAD}(V_i))$ and $E^{OPE}_{k_2}(Z_i - \text{AAD}(V_i))$, or $f(\hat{Z}_i + \text{AAD}(V_i))$ and $E^{OPE}_{k_2}(Z_i + \text{AAD}(V_i))$ would also contradict the POPF security of the OPE. Therefore, (B) contradicts the security of OPE.

We have shown that both (A) and (B) would contradict the security of the underlying schemes. Hence, assuming that the underlying schemes are secure, Sim_S is able to produce an output with a distribution indistinguishable from that of $View^{\Pi}_S(Z_i, Y_i)$, and therefore, the protocol is secure against honest-but-curious carriers. □

C Security of Protocol $\Pi^\star$

In order to formalise security against malicious adversaries, one usually compares a real execution of the protocol with an ideal execution. During the ideal execution which takes place in an ideal world, both device and carrier submit their inputs to a trusted party TP at the beginning of the protocol. TP computes the outputs of the parties and sends the outputs back to the parties. For an ideal device ID and an ideal carrier IS, let $Ideal^{\text{I}\Pi^\star}_{ID,IS}(Z_i, Y_i)$ denote the joint output of the execution of the ideal protocol $\text{I}\Pi^\star$ for computing $s_i(t)$, where Z_i is the input of ID and Y_i is the input of IS in the ideal world. Also let $Real^{\Pi^\star}_{D,S}(Z_i, Y_i)$ denote the joint output of the real device D with input Z_i and real carrier S with input Y_i after a real execution of protocol $\Pi^\star$. We use M as a prefix to denote 'malicious' and similarly H to denote 'honest'. Security of $\Pi^\star$ is defined as follows. We say $\Pi^\star$ is *perfectly secure* against malicious devices if for any malicious real-world device algorithm MD, there exists an ideal world algorithm MID such that for all Z_i and Y_i the output in the ideal world $Ideal^{\text{I}\Pi^\star}_{ID,IS}(Z_i, Y_i)$ is computationally indistinguishable from the output in the real world $Real^{\Pi^\star}_{D,S}(Z_i, Y_i)$. Perfect security is defined based on a perfect ideal-world protocol in which the trusted party TP is given all the inputs, carries out all the calculations, and outputs the score only to the carrier. This captures the ideal security requirement that the device learns nothing by participating in the protocol and the carrier only learns only the feature score.

In the case of score computing protocols however, in order to achieve higher efficiency, we do not aim for perfect security and view some information leakage acceptable. For each feature, we accept leakage of the AAD of that feature in the stored user profile to the device. We also allow the carrier to learn the order of the encrypted profile values with respect to each other. Hence we relax the above definition as follows. To incorporate the leakage of the AAD on one hand and the ordering information on the other, we model the ideal protocol $\text{I}\Pi^\star$ in the following way. After receiving each entity's reported input, i.e., the current feature reading from the device and the stored user profile from the carrier, TP calculates the score along with the AAD of the profile features and the ordering information of the new feature with respect to the profile features. Then TP outputs to the device the AAD of the profile features, and to the carrier the score and the ordering information. Considering this ideal protocol, we define security against malicious devices as follows:

Definition 3 *Let (HD, HS) and (HID, HIS) denote the honest device and carrier programs for protocol $\Pi^\star$ in the real and ideal world respectively. Let $\text{I}\Pi^\star$ be the ideal protocol in which upon receiving each party's input, the TP outputs the AAD of the carrier input to the device. We say that $\Pi^\star$ is a* ***private score computing protocol for malicious devices*** *if for any probabilistic polynomial-time algorithm MD, there exists a probabilistic polynomial-time algorithm MID such that for all Z_i, Y_i:*

$$Ideal^{\text{I}\Pi^\star}_{MID,HIS}(Z_i, Y_i) \overset{c}{\equiv} Real^{\Pi^\star}_{MD,HS}(Z_i, Y_i) \ .$$

Intuitively, the above definition guarantees that a malicious device following an arbitrary strategy does not find any information other than the AAD of the stored profile features. In the following we prove that our protocol $\Pi^\star$ satisfies this definition.

C.1 Proof of Theorem 2

Proof: (Outline) We prove the security of $\Pi^\star$ in two stages. First, we prove that the protocol is secure against malicious devices and then we prove that the protocol is secure against honest-but-curious carriers. We provide a sketch of the first stage of the proof in the following. The second stage of the proof is similar to that of Theorem 1, and hence we do not repeat it.

Stage 1. Based on Definition 3, we have to prove that for every probabilistic polynomial-time algorithm MD, there exists a probabilistic polynomial-time algorithm MID such that for all Z_i, Y_i:

$$Ideal^{\text{I}\Pi^\star}_{MID,HIS}(Z_i, Y_i) \overset{c}{\equiv} Real^{\Pi^\star}_{MD,HS}(Z_i, Y_i) \ ,$$

where Z_i, Y_i are respective inputs of the device and the carrier. We note that, as the carrier is honest, in the ideal world, HIS forwards its input Y_i without any change to TP, and hence $Ideal^{\text{I}\Pi^\star}_{MID,HIS}(Z_i, Y_i)$ will be the score produced by TP on receiving the honest input Y_i from HIS and an arbitrary value $\hat{Z}_i = MID(Z_i)$ from MID. In other words, to ensure security against a malicious device, we have to show that for any possible device behaviour in the real world, there is an input that the device provides to the TP in the ideal world, such that the score produced in the ideal world is the same as the score produced in the real world.

Given a real-world malicious device MD, the ideal world device MID is constructed as follows. MID executes MD to obtain the current encrypted feature value $E_{pk}^{HE}(\hat{Z}_i)$ and a proof of knowledge of the plaintext. By rewinding the proof, MID is able to extract $\hat{Z}_i$. MID sends $\hat{Z}_i$ to TP which replies with the AAD of the HIS input: AAD(V_i).

MID then selects ℓ_i arbitrary values to construct a mock user profile such that the AAD of the mock profile features is equal to AAD(V_i). It then is able to calculate all the following values according to the protocol for all j and k: $E_{pk}^{HE}(\delta_{ij}^{\text{L}})$, $E_{pk}^{HE}(\delta_{ij})$, $E_{pk}^{HE}(\delta_{ij}^{\text{H}})$, $E_{pk}^{HE}(\delta'_{ijk})$, $E_{pk}^{HE}(\delta'^{\text{L}}_{ijk}))$, and $E_{pk}^{HE}(\delta'^{\text{H}}_{ijk})$. MID shuffles these values and sends them to MD. The latter three sets of values are distributed identically to the protocol. The former three sets of values are based on mock profile feature values rather than the real ones. However, the malicious device MD is assumed to have no knowledge about the user behaviour, and hence the device is not able to distinguish them hidden within the latter three sets of values. MD then replies and indicates for each one of the received values if they are positive, negative, or zero. MID checks all the values and makes sure MD does not cheat. MD does not get any output at the end of the simulation.

From all the values MD receives $\frac{1}{\sigma+1}$ of them deviate from a real protocol execution, however these values are hidden within the values that are calculated following the protocol. Thus, MD has at most $\frac{1}{\sigma+1}$ chance of being able to output a value which would be distinguishable from a real world execution of $\Pi^\star$. This means that the protocol is secure against malicious devices with probability at least $\frac{\sigma}{\sigma+1}$.

Stage 2. With similar arguments presented in the proof of Theorem 1, we can claim that a honest-but-curious carrier only learns the order of the feature data. Therefore, we can similarly show that protocol $\Pi^\star$ is secure against an honest-but-curious carrier. $\square$