HCFContext: Smartphone Context Inference via Sequential History-based Collaborative Filtering

Vidyasagar Sadhu*, Saman Zonouz*, Vincent Sritapan†, and Dario Pompili*
*Department of Electrical and Computer Engineering, Rutgers University, New Brunswick, USA
†Cyber Security Division, Department of Homeland Security Science & Technology Directorate, USA
*{vidyasagar.sadhu, saman.zonouz, pompili}@rutgers.edu, †vincent.sritapan@hq.dhs.gov

***Abstract*—Mobile context determination is an important step for many context-aware services such as location-based services, enterprise policy enforcement, building/room occupancy detection for power/HVAC operation, etc. Especially in enterprise scenarios where policies (e.g., attending a confidential meeting only when the user is in "Location X") are defined based on mobile context, it is paramount to verify the accuracy of the mobile context. To this end, two stochastic models based on the theory of Hidden Markov Models (HMMs) to obtain mobile context are proposed—*personalized model* (*HPContext*) and *collaborative filtering model* (*HCFContext*). The former predicts the current context using sequential history of the user's past context observations; the latter enhances *HPContext* with collaborative filtering features, which enables it to predict the current context of the primary user based on the context observations of users related to the primary user, e.g., same team colleagues in company, gym friends, family members, etc. Each of the proposed models can also be used to enhance/complement the context obtained from sensors. Furthermore, since privacy is a concern in collaborative filtering, a privacy-preserving method is proposed to derive *HCFContext* model parameters based on the concepts of homomorphic encryption. Finally, these models are thoroughly validated on a real-life dataset.**

***Index Terms*—Mobile context, collaborative filtering, privacy-preserving, personalized model, sensors, location, prediction.**

I. Introduction

Overview: Mobile device applications provide an increasing number of features customized to match users' needs. These needs are very often inferred from specific features such as the user location, activity (e.g., running, walking, driving), surrounding people, interacting people, the current app usage on the device, etc. These features collectively define a specific user (mobile) *context*. Mobile applications are increasingly making use of these contexts such as location-based services (e.g., Foursquare, Google Now, Weather updates, etc.), enhanced reality applications (Pokemon GO [1]), continuous authentication, etc. However, to enable these services, context inference is a much needed and important step. *Most of the existing work focuses on obtaining mobile context instantaneously from sensors which could possibly be hacked, noisy or insufficient and as such cannot be relied in certain security applications. Hence we take a different approach to that problem in this paper by modeling mobile context based on past context data.* There are many advantages of modeling the user context by leveraging the sequential nature of context information in a user's history as it can be used to predict the current or future contexts. The former can be used to validate and/or enhance the possibly hacked/noisy/insufficient sensor context, while the latter can provide some information ahead of time to the benefit of the user [2]. For example, a user's general routine during weekdays could be to head first to Starbucks near his home, then to his work and then to Gym and back to home as shown in Fig. 1(left). *The learned model will capture this behavior and can be used to validate the location of the user obtained via GPS at 5:30 pm to be at Gym (current context prediction) or display coupons related to Starbucks on his phone in advance (future context prediction).* The latter can be leveraged by mobile personal assistant technologies such as Apple Siri/Google Now to much benefit of the user.

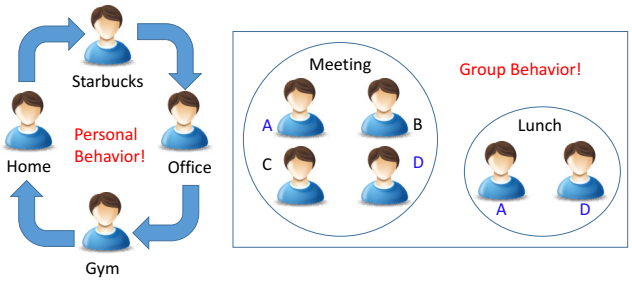


Fig. 1. A user's personal (left) and group/collaborative filtering (right) behavior that can provide some clues about user's context at any given time.

Motivation: One of the context-aware services is the enterprise data access control as in [3], where policies are defined for enterprise data access based on the phone's context (e.g., connected Wi-Fi, Cell ID, time, etc.). For example, a policy may be defined to allow the phone to be used to attend a confidential meeting or open a confidential document only when the phone's context is found to be within a given location and time. In these secure scenarios, it is not suggested to rely solely on the context obtained from the phone's sensors (e.g., GPS/WiFi and System clock to give location and time) because *they could be hacked unknowingly to the user.* For example, a virus might change the system clock to show different time or spoof the GPS [4] to show different location. However it is hard to hack a model (more so, a collaborative one) that is learned over a long period of time. Hence our solution can be used to *validate* the context directly obtained from sensors at that instant. Secondly, it is possible that context from sensors is *noisy (due to malfunction) or does not contain enough information*. For example, in the case of a tablet or old mobile phone, it may not be able to acquire (accurate) GPS signal. As such it will be helpful if there is another way of obtaining this information such that it complements the context from sensors. Thirdly, as mentioned earlier, future context prediction is useful in certain context-aware services such as mobile personal assistant technologies (Google Now, Apple Siri, etc.), which can help *pre-fetch information/pre-plan based on the predicted context*.

Our Approach: In order to address the above issues, we first propose a personalized model (*HPContext*) that predicts

978-1-5386-9148-9/19/$31.00 ©2019 IEEE

the user's context based on its past sequential history of contexts. Obtaining context through two approaches—sensors and personalized model—adds an extra layer of confidence to the obtained context. However, the following situations are possible: contexts obtained from both approaches are very different, contexts from one of the approaches is not available (e.g., GPS may not be available from phone sensors indoors, etc.) or insufficient leading to uncertainty. In such situations, assuming the user is closely connected to a group of people (e.g., same team colleagues in the company as shown in Fig. 1(right), gym friends, family members, etc.), *it is possible that the context of other members in that group of people can provide additional information about his/her context.* For example, assume users A and D often go to lunch together (learned via model). Now somehow if it is known that D is going to "Restaurant1" tomorrow for lunch, it is most likely that the context of user A tomorrow around 1 pm is "having lunch with D at Restaurant1" without having to rely on A's phone sensors at that instant. Our paper explores this aspect of context to provide a second layer of confidence to the context (over and above the personalized model). For this purpose, we propose to use such context obtained through *collaborative filtering* of the contexts of users closely related to the primary user (*HCFContext*). To the authors' best knowledge, this is the first work to explore collaborative filtering for mobile contexts that can be used to *validate* and/or *enhance* the *current* context obtained from sensors or predict the *future* context for mobile personal assistant technologies. Additionally we present a privacy-preserving method for parameter estimation (training) of *HCFContext*, as users may not be willing to share their private data with each other for the same.

Contributions: Our specific contributions are as follows.

- We propose a personalized (*HPContext*) and collaborative filtering (*HCFContext*) model to *predict* the users context at any given instant (including future) based on the sequential history of past contexts and based on the theory of Hidden Markov Models (HMMs). We design a novel emission model for these HMMs by considering the unique features and the nitty-gritties of a mobile context (e.g., GPS from sensors may not be always available).
- We present a homomorphic encryption based privacy-preserving approach for training the *HCFContext*.
- We validate the efficacy of the proposed models by testing them on a real-life data set belonging to five graduate students collected over two months. We also evaluate our privacy-preserving approach to study its trade-offs.

Paper Outline: In Sect. II, we present the related work and position our paper. In Sect. III, we discuss the proposed models, (*HPContext*, *HCFContext*) and the privacy-preserving approach for the parameter estimation of *HCFContext*. In Sect. IV, we present the results of our proposed approaches. Finally, in Sect. V, we conclude and discuss future directions.

II. Related Work

In this section, we position our work with respect to previous works (i) that obtain context with and without users sequential history, (ii) via local collaborative sensing, and (iii) related to privacy-preserving collaborative filtering.

Without Sequential History: There is existing work on modeling mobile contexts without considering the sequential nature of context information. For example, Bao et al. [5] propose an unsupervised approach to model mobile context from raw contextual data using Latent Dirichlet Allocation (LDA). Srinivasan et al. [6] mine the co-occurrences of certain context attributes; frequently and simultaneously occurring context attributes are formulated as association rules to predict what else the user will do (e.g., read comics) given a current context attribute (e.g., listen to jazz). However, unlike ours these approaches do not exploit temporal dependencies among contexts but only consider the behavior at a given instant.

With Sequential History: There is also work that exploits the sequential/temporal dependencies between contexts. For example, Mukherji et al. [7] present Mobile Sequence Miner (MSM) framework that mines frequent sequences occurring in app usage patterns, location visits, and call logs using a frequency-based approach. Farrahi et al. [8] present a probabilistic approach to mine mobile phone data (e.g., location) sequences using Distant N-Gram Topic Model (DNTM) where they model the sequence to be dependent on the starting element of the sequence. There are works that model the user activity using HMM based on sensor measurements [9], [10]. Even though these approaches exploit the sequential nature of contextual information, they neither consider collaborative filtering nor privacy-preserving aspects like we do. We claim that collaborative filtering context has additional context information than context obtained from personal history alone.

Local Collaborative Sensing: There are also works on local collaborative sensing; however, these works do not consider the sequential nature of past information into the collaboration process [11]. For example, Mantyjarvi et al. [12] present a collaborative sensing approach where a device, upon noticing a change in its local context beyond a threshold value, requests contexts from its surrounding devices so as to increase the accuracy of its context vector. Miluzzo et al. [13] use collaboration to increase the confidence of the sensed context through consensus of contexts sensed at surrounding devices. These approaches however do not consider past sequential nature of context information into collaboration.

Privacy-preserving Collaborative Filtering: There is existing literature in the domain of privacy-preserving collaborative filtering and HMM techniques, which can be broadly classified into two categories—*data perturbation/randomization* to hide the original data albeit with accuracy loss and *data encryption* with typically no accuracy loss albeit with higher computational complexity. On the former, Polat et al. [14] and Parameswaran et al. [15] present privacy-preserving collaborative filtering techniques based on randomized perturbation and data obfuscation respectively. On the latter, Guo et al. [16] present a privacy-preserving Markov model for sequence classification using homomorphic and ElGamal cryptographic systems. More works in this category can be found in [17]–[19]. We present an approach, designed specifically for our scenario, that extends the ideas in this category for privacy preserving multi-party parameter estimation of our *HCFContext* model.

III. Proposed Approach

In this section, we describe *HPContext* and *HCFContext* models (Sect. III-A) and a privacy-preserving approach for parameter estimation of *HCFContext* (Sect. III-B).

A. Context from Collaborative Filtering (HCFContext)

Problem Formulation: We present here the proposed *HCFContext* model *by designing a novel emission model of a HMM taking into account the multi-user collaborative filtering*

TABLE I
EXAMPLE CONTEXT OBSERVATIONS OF USER u FOR TIMES $t = 1, ..., T$.

Time (t)	**Context observation of user u at time t ($\boldsymbol{O}_{tu}$)**
t_1	WiFi: *wifi1*, CellID: *cid1*, LAC: *lac1*, Battery Level: *high*, Battery Status: *discharging*, Day Period: *morning*, Day of week: *Monday*, Holiday: *No*
t_2	WiFi: *wifi2*, CellID: *cid2*, LAC: *lac2*, Battery Level: *medium*, Battery Status: *discharging*, Day Period: *noon*, Day of week: *Monday*, Holiday: *No*
.....	
t_T	WiFi: *wifi1*, CellID: *cid1*, LAC: *lac1*, Battery Level: *low*, Battery Status: *charging*, Day Period: *night*, Day of week: *Sunday*, Holiday: *Yes*

aspects, as well as the unique features of the mobile context and its nitty-gritties such as feature unavailability. We first start with notation—capital letters denote random variables, whereas their small equivalents are their realizations. A vector variable will be indicated in bold. We model context (C_t) as a latent variable in the HMM. For a given user, the observation corresponding to a context state at time t will be called context observations ($\boldsymbol{O}_t$). An example of a user's context observations from time $t = 1...T$ is shown in Table I. Each observation, $\boldsymbol{O}_t$, consists of a set of contextual feature-value pairs. These observations are obtained at regular time intervals (e.g., a minute to four hours). It can be seen as an example from the table that the context observation at $t = t_1$ corresponds to morning when the user is at home (battery is high, probably because the user charges her phone the previous night). The observation at $t = t_2$, say after 4 hours, can be interpreted as being at office or workplace (change of WiFi, Cell ID, etc.) with battery level being in medium range. Finally, the observation at $t = t_T$ (after several days) can be taken to be again at home in the night. We assume that K number of latent context states spans across these T observations. Considering users $u = 1, ..., M$, each user has a similar set of T observations. Plate notation [20] for our *HCFContext* model for M users is shown in Fig. 2. In plate notation, the number of different categorical values a random variable can take is shown inside the circle or rectangle. A circle is used for a random variable while a rectangle is generally used for hyperparameters. Observable variables are shaded. The number of repetitions of a rectangular block is shown at its bottom right corner. For a given user u, the observation at time t, $\boldsymbol{O}_{tu}$ is a set of feature-value pairs (as in a row of Table I). We can write $\boldsymbol{O}_{tu} = (\boldsymbol{f}_{tu}, \boldsymbol{v}_{tu}) = (f_{t,u,i}, v_{t,u,i})_{i=1}^{|f_{tu}|}$, where $|f_{tu}|$ is the number of available features of user u, at time t. Generation of each variable in Fig. 2 is described next.

Initial State Model: A prior distribution of *contexts*, π is generated from prior Dirichlet distribution, η. C_1 is then generated from π. We will assume a total of K possible context states for *HCFContext* over all M users.

Transition Model: A prior *transition* distribution of contexts, $\rho_{c_{t-1}} = \rho_k$, is generated from a prior Dirichlet distribution, ω_k. C_t is then generated from $\rho_{c_{t-1}}$ for a given C_{t-1}. Note that ρ_k and ω_k can take a total of K categorical values (C_t, current state) for each k (C_{t-1}, previous state).

Novel Emission Model: For $\boldsymbol{O}_t$ generation under each C_t, since features are not always available (e.g., GPS is not available when indoor or underground, etc.), we will define a separate distribution for features (F_t) to account for their availability and then another distribution to obtain the values (V_t) for those features at time t, as illustrated in Fig. 2. F_t is dependent on C_t, whereas V_t is dependent on both C_t and F_t.

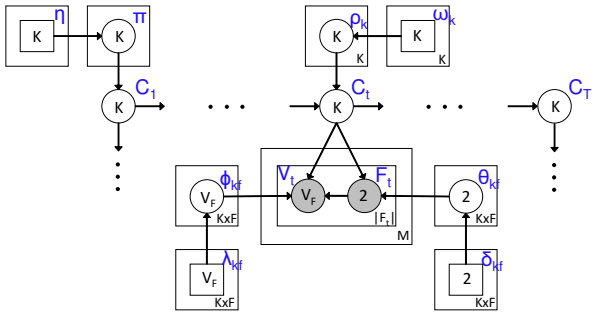


Fig. 2. Plate notation of *HCFContext*. F is the total number of features, V_F denotes the total number of possible values for feature F, $|F_t| = |V_t|$ denotes the number of features observed at time t, M is the number of users, and K is the number of hidden context states.

An initial *feature* distribution, $\theta_{c_t,f_t} = \theta_{k,f}$, is generated from a prior Dirichlet distribution, $\delta_{k,f}$. Feature F_t is then generated from $\theta_{k,f}$, which can take two categorical values—whether the feature is present or not for the given context, c_k. We will assume a total of F possible features over all observations of all users. A prior *value* distribution, $\phi_{c_t,f_t} = \phi_{k,f}$, is generated from a prior Dirichlet distribution, $\lambda_{k,f}$. Value V_t is then generated from ϕ_{c_t,f_t} for a given context c_t and feature $f_{t,i}$. For a given feature f, we will assume V can take V_f possible values. The priors will be chosen such that the summation and non-negativity constraints on the parameters are satisfied and also to encode prior information. For example, if it is known that a user frequently moves between home to work, the prior parameter for this transition, (ω_k), is given a high value.

Parameter Estimation (Training): Given these parameters, $\boldsymbol{\Psi} = \{\Pi, P, \Theta, \Phi\}$, the parameter space of $\pi, \rho_k, \theta_k, \phi_{k,f}$ and their hyperparameters, $\{\eta, \omega, \delta, \lambda\}$, let us denote the joint probability of all latent contexts $\boldsymbol{C} = \{C_1, ...C_T\}$ and all context observations, $\boldsymbol{O} = \{O_1, ...O_T\}$, as $P(\boldsymbol{C}, \boldsymbol{O} \mid \boldsymbol{\Psi}, \eta, \omega, \delta, \lambda)$. Likelihood of the observations $\boldsymbol{O}$ is then,

$$L(\boldsymbol{O}) = \sum_{\boldsymbol{c}} P(\boldsymbol{C}, \boldsymbol{O} \mid \boldsymbol{\Psi}, \eta, \omega, \delta, \lambda). \tag{1}$$

Training the HMM involves finding the parameters $\boldsymbol{\Psi}$ that maximize the likelihood in (1). Given the complex nature of (1) expanded, it is very difficult to derive a closed-form solution of $\boldsymbol{\Psi}$. Hence, we make use of the well-known iterative approach called Expectation Maximization (EM) [21] *but modify it to fit our approach*. It consists of two important relations viz., forward and backward relations. We can express the full observation probability $p(\boldsymbol{o}_t \mid c_{tk}) = \mu_{tk}$:

$$\mu_{tk} = p(\boldsymbol{o}_t \mid c_{tk}) = \prod_{u=1}^{M} \mu_{tk_u} = \prod_{u=1}^{M} \prod_{\substack{f \in \boldsymbol{f}_{tu} \\ v \in \boldsymbol{v}_{tu}}} \theta_{k,f}\ \phi_{k,f,v}. \tag{2}$$

Forward relation can now be expressed as,

$$\alpha(c_{1k}) = \pi_k \mu_{1k} \text{ (for t = 1)}, \tag{3}$$

$$\alpha(c_{tk}) = \mu_{tk} \sum_{j=1}^{K} \alpha(c_{t-1,j}) \rho_{jk} \text{ (for t = 2 to T)}. \tag{4}$$

Similarly, we can express the backward relation as follows,

$$\beta(c_{tk}) = \sum_{j=1}^{K} \beta(c_{t+1,j}) \mu_{t+1,j} \rho_{kj}. \tag{5}$$

Using these relations, we can define two new variables, $\xi(C_{t-1}, C_t)$ and $\gamma(C_t)$, for ease of analysis (denote $\gamma(C_t = c_{tk}) = \gamma(c_{tk})$ and similarly for $\xi(c_{t-1,j}, c_{tk})$) as follows,

$$\xi(c_{t-1,j,tk}) = \frac{\alpha(c_{t-1,j})\mu_{tk}\rho_{jk}\beta(c_{tk})}{\sum_{k=1}^{K}\alpha(c_{Tk})}, \tag{6}$$

$$\gamma(c_{tk}) = \sum_{j=1}^{K}\xi(c_{t-1,j}, c_{tk}). \tag{7}$$

We can now compute the model parameters as,

$$\pi_k = \frac{\gamma(c_{1k}) + \eta_k}{\sum_{k'=1}^{K}(\gamma(c_{1k'}) + \eta'_k)}, \tag{8}$$

$$\rho_{kj} = \frac{\sum_{t=2}^{T}\xi(c_{t-1,k}, c_{tj}) + \omega_{kj}}{\sum_{j'=1}^{K}\sum_{t=2}^{T}\xi(c_{t-1,k}, c_{tj'}) + \sum_{j'=1}^{K}\omega_{kj'}}, \tag{9}$$

$$\theta_{k,f} = \frac{\sum_{t=1}^{T}\gamma(c_{tk})\sum_{u=1}^{M}\boldsymbol{I}(f \in \boldsymbol{f}_{tu}) + \delta_{k,f}}{M\sum_{t=1}^{T}\gamma(c_{tk}) + \sum_{f'=1}^{F}\delta_{k,f'}}, \tag{10}$$

$$\phi_{k,f,v} = \frac{\sum_{u=1}^{M}\sum_{t:f\in\boldsymbol{f}_{tu}}\gamma(c_{tk})\boldsymbol{I}(v_{t,f,u} = v) + \lambda_{k,f,v}}{\sum_{u=1}^{M}\sum_{t:f\in\boldsymbol{f}_{tu}}\gamma(c_{tk}) + \sum_{v'=1}^{V_f}\lambda_{k,f,v'}}, \tag{11}$$

where $\boldsymbol{I}(x)$ is the indicator function with $\boldsymbol{I}(x) = 1$ if x is true and 0 otherwise. Eqs. (3)-(7) constitute the E-step, while Eqs. (8)-(11) constitute the M-step of the EM algorithm. These steps are iterated until the parameters in M-step converge.

Prediction: We will now use the learned parameters to predict the future observations given past observations. This will be done by first finding the distribution over future states and then by multiplying the distribution over the observations given the future state. In our case, since the observations are feature-value pairs, we will first calculate the distribution over features and then the distribution over values given features. Given that the user has made a sequence of past 't' observations, $\boldsymbol{o}_{1:t} = \{\boldsymbol{o}_1, ..., \boldsymbol{o}_t\}$, the probability that a feature f, and then a value v for that feature, will be observed at time $t+1$ can be computed, respectively, as follows,

$$p(f \in \boldsymbol{f}_{t+1} \mid \boldsymbol{o}_{1:t}) = \sum_{k=1}^{K} p(c_{t+1,k} \mid \boldsymbol{o}_{1:t}) \cdot \theta_{k,f}, \tag{12}$$

$$p(v_{t+1,f} = v \mid \boldsymbol{o}_{1:t}) = \sum_{k=1}^{K} p(c_{t+1,k} \mid \boldsymbol{o}_{1:t}) \cdot \phi_{k,f,v}, \tag{13}$$

Note that $p(c_{t+1,k} \mid \boldsymbol{o}_{1:t})$ in (12), (13) is computed as,

$$p(c_{tk} \mid \boldsymbol{o}_{1:t}) = \sum_{j=1}^{K} p(c_{t+1,k} \mid c_{tj}) p(c_{tj} \mid \boldsymbol{o}_{1:t}), \tag{14}$$

where $p(c_{t+1,k} \mid c_{tj}) = \rho_{jk}$ and $p(c_{tj} \mid \boldsymbol{o}_{1:t})$ can be recursively computed using a procedure similar to (4),

$$\alpha'(c_{tk}) = p(c_{tk} \mid \boldsymbol{o}_{1:t}) = \frac{\mu_{tk}\sum_{j=1}^{K}\rho_{jk}\alpha'(c_{t-1,j})}{\sum_{k=1}^{K}\mu_{tk}\sum_{j=1}^{K}\rho_{jk}\alpha'(c_{t-1,j})}$$

We compute (12), (13) over all features, $f_i : i = \{1, ..., F\}$, all corresponding values $v_j : j = \{1, ..., V_{f_i}\}$ and pick the most probable ones to get the feature-value pairs at $t+1$.

Determining the Number of Hidden Contexts: So far we have assumed that the number of hidden contexts K is given; however, in general this number needs to be determined automatically from the data. We will now detail an original approach to determine the best K assuming it lies in the range $\mathcal{K}_{range} = [K_{min}, K_{max}]$ and that the extremes can be approximately obtained from prior information about the data. To determine the best $K \in \mathcal{K}_{range}$, we will define a metric called *Perplexity* that determines how well the chosen K fits for prediction tasks over the testing set. Finding perplexity involves calculating the prediction probability over a sequence of observations given a sequence of past observations from the testing set. Perplexity can be defined as follows,

$$\text{Perplexity} = \exp\left(-\frac{\log p(\boldsymbol{o}_{t+1:T} \mid \boldsymbol{o}_{1:t})}{\sum_{u=1}^{M}\sum_{t=t+1}^{T}|\boldsymbol{o}_{tu}|}\right), \tag{15}$$

where $|\boldsymbol{o}_{tu}|$ is the number of features observed at time t for user u and $p(\boldsymbol{o}_{t+1:T} \mid \boldsymbol{o}_{1:t})$ can be determined as follows,

$$p(\boldsymbol{o}_{t+1:T} \mid \boldsymbol{o}_{1:t}) = \sum_{k=1}^{K} p(c_{tk} \mid \boldsymbol{o}_{1:t}) p(\boldsymbol{o}_{t+1:T} \mid c_{tk}), \tag{16}$$

where $p(c_{tk} \mid \boldsymbol{o}_{1:t}) = \alpha'(c_{tk})$ and $p(\boldsymbol{o}_{t+1:T} \mid c_{tk}) = \beta(c_{tk})$. Intuitively, a small perplexity is desired. Although in general the perplexity reduces as K increases, a large K is not preferred due to the risk of overfitting. Hence, we make use of the rate of decrease in perplexity to determine when to stop increasing K, e.g., if it falls below a threshold (say 10%).

***HPContext* Model:** Personalized model is just a special case of the above collaborative filtering model when the number of users is one (i.e., $M = 1$). The main difference is in the observation probability. Specifically, for user u,

$$\mu_{tk} = p(\boldsymbol{o}_{tu} \mid c_{tk}) = \mu_{tk_u} = \prod_{\substack{f\in\boldsymbol{f}_{tu}\\ v\in\boldsymbol{v}_{tu}}} \theta_{k,f}\ \phi_{k,f,v}. \tag{17}$$

Remaining equations remain the same with setting $M = 1$.

B. Privacy-preserving Multi-party Computing

Since the users could possibly be in different contexts while training it is important to preserve their privacy. In this section, we develop algorithms for multi-party parameter estimation of *HCFContext* (Sect. III-A) while preserving each party's privacy. Hence model parameters need to be jointly estimated when the individual observations are encrypted. Since the model parameters $\boldsymbol{\Psi}$ are known to everyone, the prediction can be carried out by each party on their own. The algorithms we developed for this (extended from [19]) are based on the following well-known primitives—*homomorphic encryption* [22], *secure logsum*, and *secure negation* [23].

The proposed algorithm for secure multi-party computation of data likelihood, $P(\boldsymbol{O} \mid \boldsymbol{\Psi})$ is shown in self-explanatory Algorithm 1. It details the steps the M parties need to undertake to jointly compute $\alpha(\cdot)$ (as per (3), (4)) and the log likelihood of their observation data given the model parameters, $P(\boldsymbol{O} \mid \boldsymbol{\Psi})$. We assume only one party (P_1 in Algo. 1) has both the private and public keys, while the remaining $M-1$ parties ($P_m \mid m = 2, \ldots, M$) have only the public key. Hence all parties encrypt their private observation data and send it to P_m, where m can be any one of $2, \ldots, M$, which does the computations on this private encrypted data (as it cannot decrypt it since it has only public key). Whenever it needs to compute secure logsum and secure negation protocols, it consults P_1 which has the private key. Algo. 1 can be similarly

Algorithm 1 Secure Multi-party Computation of $P(\boldsymbol{O} \mid \boldsymbol{\Psi})$.

Input: Parties $P_1, P_2, \ldots, P_M$ know the model $\boldsymbol{\Psi}$. Each party has a set of private observations $\boldsymbol{O}_{tu}=(\boldsymbol{f}_{1u}, \boldsymbol{v}_{1u}), (\boldsymbol{f}_{2u}, \boldsymbol{v}_{2u}), \ldots, (\boldsymbol{f}_{Tu}, \boldsymbol{v}_{Tu})$.
Output: $P(\boldsymbol{O} \mid \boldsymbol{\Psi}, \eta, \omega, \delta, \lambda) = \sum_{k=1}^{K} \alpha(c_{Tk})$

Initialization (t=1):
for $k = 1, \ldots, K$ **do**
  for all $P_{q \neq m}$ **do**
    P_q sends $E[\log(\mu_{1k_x})]$ to P_m,
  end for
  P_m computes $E[\log(\prod_{u=1}^{M} \mu_{1k_u})]$ using (2)
  P_m computes $E[\log(\alpha(c_{1k}))]$ using (3)
end for

Induction:
for $t = 2, \ldots, T$ **do**
  Repeat steps 2-6, replacing time index with t, so that P_m obtains $E[\log(\prod_{u=1}^{M} \mu_{tj_u})]$ for $j = 1, \ldots, K$.
  for $k = 1, \ldots, K$ **do**
    P_m and P_1 use the secure logsum protocol to compute $E[\log \sum_{j=1}^{K} \alpha(c_{t-1,j})\rho_{jk}]$,
    P_m computes $E[\log(\alpha(c_{tk}))]$ using (4)
  end for
end for

Termination:
P_m and P_1 use the secure logsum protocol to compute $E[\log P(\boldsymbol{O} \mid \boldsymbol{\Psi})] = E[\log \sum_{k=1}^{K} \alpha(c_{Tk})]$,
P_1 decrypts the result and sends the value to P_m.

Algorithm 2 Secure Multi-party Estimation of $\boldsymbol{\Psi}$.

Input: $E[\log \alpha(c_t)]$, $E[\log \beta(c_t)]$, and $E[\log P(\boldsymbol{O} \mid \boldsymbol{\Psi})]$.
Output: The updated model parameters $\boldsymbol{\Psi} = \{\Pi, P, \Theta, \Phi\}$
for $t = 2, \ldots, T$ **do**
  for $k = 1, \ldots, K$ **do**
    for $j = 1, \ldots, K$ **do**
      P_m computes $E[\log \xi(c_{t-1,j}, c_{tk})]$ by taking the log of (6)
    end for
    P_m and P_1 use the secure logsum to compute $E[\log \gamma(c_{tk})]$ from (7)
  end for
end for
P_m uses (8), (9) to update $E[\log \pi_k]$, $E[\log \rho_{kj}]$.
P_m then updates $E[\log \theta_{k,f}]$ and $E[\log \phi_{k,f,v}]$ as in (10) and (11), for the M parties using the secure logsum and negation protocols.

used to compute $\beta(\cdot)$ as per (5). The proposed algorithm for secure multi-party estimation of model parameters, $\boldsymbol{\Psi}$ is shown in Algo. 2. It details the steps taken by the M parties to estimate $\boldsymbol{\Psi}$ using $P(\boldsymbol{O} \mid \boldsymbol{\Psi}), \alpha(\cdot), \beta(\cdot)$ computed from Algo. 1. In Algo. 2, P_m first computes $\xi(c_{t-1,j}, c_{tk}), \gamma(c_{tk})$ as per (6) and (7) respectively. It then computes the model parameters, $\boldsymbol{\Psi}$ as per eqs. (8) to (11). The computational complexity of Algo. 1 can be seen as $O(MK^2T)$ due to twice-nested for-loop operating for T timesteps for each party, while it is $O(MK^3T)$ for Algo. 2 due to triple-nested for-loop.

Threat/Adversary Model: We use a semi-honest setting, where parties keep all their intermediate computations private, and we assume that P_m will not collude with P_1 and disclose encrypted values received. The key generation in our security model will be following a standard key exchange mechanism [24] without the need to a third party entity. One may argue that the models themselves may be unreliable as they are based on historical sensor data. Our approach provides protection against this point by comparing sensor context with that obtained from *HPContext* and *HCFContext*, and alerting the user in case of significant differences and adaptively learning to ignore false positives.

Floating Point and Negative Numbers: Our algorithms need to encrypt the HMM parameters, which are real numbers. We translate between floating-point numbers and non-negative integers by scaling and rounding off the values. Let c be the scaling factor. A real number r is translated to integer $\bar{r} = \lfloor cr \rfloor$, where $\lfloor x \rfloor$ is the largest integer $\leq x$. We incorporate this operation into the encryption and decryption as $E'[r] = E[\bar{r}] = E[\lfloor cr \rfloor]$, *and* $D'[E'[r]] = \bar{r}/c \approx r$. For negative numbers, we use modulo n arithmetic, i.e., negative numbers are represented by their modular additive inverse. For $r < 0$, $E'[\bar{r}] = E'[\bar{r}+n]$. This means our r is limited to range $[-n/(2c), (n-1)/(2c)]$.

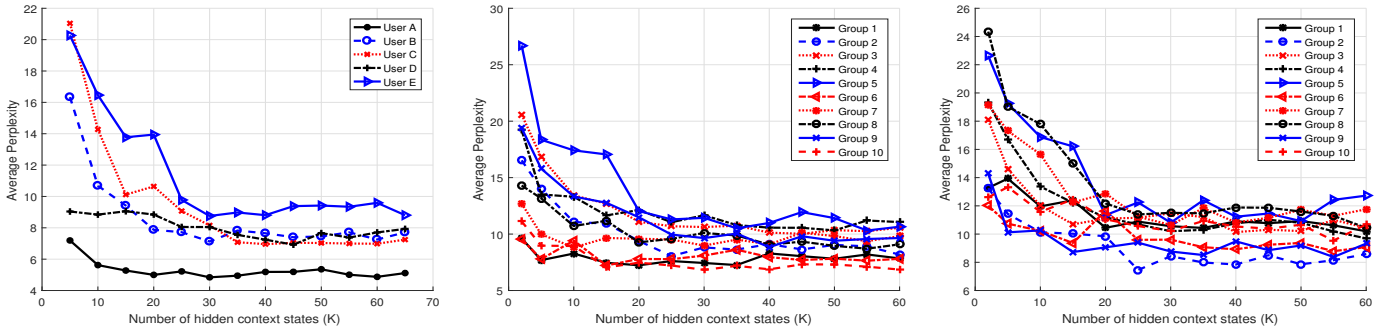
TABLE II
LIFEMAP DATASET ANALYSIS (9 WEEKS, 5 USERS, 1 HR SAMPLING).

Feature	Values	V_F
WiFi	MAC values of max dBm Access Points	440
Place Name	User defined place name values	168
Cell ID	Cell ID values	316
LAC	Location Area Code values	33
Batt. Level	Low ($< 35\%$), Medium ($35\%-65\%$), High ($65\% - 85\%$), Full ($> 85\%$)	4
Batt. Status	Charging, Discharging, Full	4
Day Period	Morning: {7 to 11 am}; Noon: {11 am to 2 pm}; Afternoon: {2 to 6 pm}; Evening: {6 to 9 pm}; Night: {9 pm to 7 am}	5
Day Name	Mon, Tue, Wed, Thu, Fri, Sat, Sun	7
Holiday	Yes, No	2

IV. EXPERIMENTAL EVALUATION

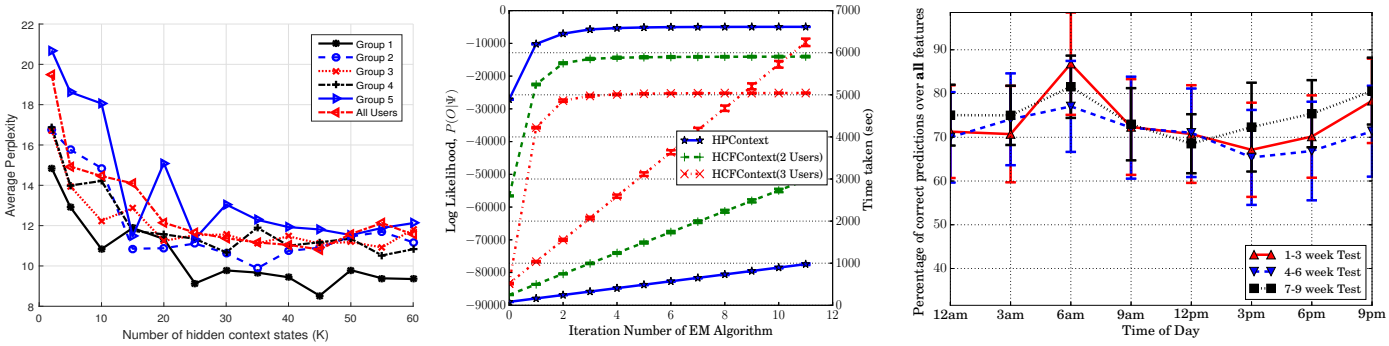
We describe the experimental setup and results, and evaluate the performance of the privacy-preserving algorithms.

Dataset and Experiment Description: To validate our models, we have used the LifeMap dataset [25], which is freely available online. This dataset consists of fine-grained mobility data such as WiFi fingerprints (MAC address and signal strengths of surrounding Wi-Fi APs), user-defined types of places (workplace, cafeteria, etc.), cell tower ID, etc. These details of 10 users are logged every 2 to 5 minutes for about two months (which is the overlap time among all users) in Seoul, Korea. The users are graduate students of the same lab in the university and as such the data suits our application. We have chosen data corresponding to five users for a period of nine weeks for our experiments. Username, gender and the number of places visited in total for these five users are as follows—*GS2 (A)*, M, 163; *GS3 (B)*, M, 297; *GS4 (C)*, F, 209; *GS7 (D)*, M, 289; *GS12 (E)*, M, 376. The average number of places visited per user is about 270. However, the number of frequently visited places for each user ranges from 8 to 35 (median 20). We also down sampled the data to 1 hour period to ease the computations i.e., the time instants t and $t+1$ are separated by 1 hour. This also means the trained models will be able to capture mostly only those contexts with stay duration of the order of an hour or more. Six weeks of data is used for training and the rest three week data is used for testing. The list of features we have used from this dataset is shown in Table II. Here V_F corresponds to the total number of values taken by that feature. In case of 'Holiday', Saturday, Sunday and any public holidays are considered as holidays. In cases where a certain feature's value is missing, we model it as if the feature is not available at that time using $\theta_{k,f}$. We have empirically set $\eta = \{1/K, ..1/K\}, \omega_k = \{50/K, ...50/K\}, \delta = \{1, 10\}, \lambda = \{0.01, ...0.01\}$ in our experiments [26] (K is the number of hidden context states). All our results (implemented in Python) are generated on an Intel 4-core i7-2600 CPU @ 3.40GHz, with 8 GB of RAM. *We did not consider a larger dataset as our main idea is to apply collaborative filtering across the user's closely related users such as labmates, roommates, etc. Also, we could not find larger datasets with similar features.*



(a) (b) (c)

Fig. 3. Perplexity vs. K (a) 1 user case, (b) 2 user case, (c) 3 user case. These figures help identify the best 2 or 3 user groups (most related users).



(a) (b) (c)

Fig. 4. (a) Perplexity vs. K (4 and all user case). (b) Log likelihood of the training data and time taken vs. number of iterations of EM algorithm for different models. (c) Comparison of performance (*HPContext* + *HCFContext* (2) + *HCFContext* (3)) for different choices of test data - first, mid, and last 3 weeks.

Perplexity vs. K and Optimal Group Selection: Figure 3a shows the averaged perplexity values for different number of hidden states, K, for one user case. Since this is a one user case it corresponds to *HPContext*. The perplexity is calculated by predicting the observations at the next 12 time instants in the test data given the preceding 12 observations on the same day. This has been done over all the days in the three week test data and the results are averaged. From Fig. 3a, we can observe that perplexity reduces as K increases with a pattern of diminishing returns. We can observe that user A has the best perplexity and $K = 10$ provides a good balance between perplexity and complexity introduced due to higher K values. Figure 3b shows similar result for 2 users. Since we have 2 users, we have a total of 10-user groups (5C2 combinations). We can observe that Group 10 consisting of users A, C has the best perplexity and $K = 15$ provides a good tradeoff. This means that the two users in Group 10 have more similar patterns than the users in other groups. Figure 3c shows similar result for 3 users. We can observe that Group 2 (users A, C, D) has the best perplexity with $K = 25$ providing good tradeoff. Similarly, for 4 user case, we can observe from Fig. 4a, that $Group\ 1$ (users A, B, C, D) has best perplexity values and $K = 25$ provides a good balance. For all user case in the same figure, we can see that $K = 25$ provides a good tradeoff. *Hence these results can be used to select most related users in a group of 2/3/4, etc. Even though it takes time to find the optimal group via this approach, we feel it is acceptable as it is done only once offline.* For the results below, whenever a user or a group of users is mentioned, we considered the above *best* groups and optimal K values. To illustrate the benefits of collaborative filtering contexts, we considered only 2 and 3 user groups as benefits diminish with increase in group size.

Log Likelihood Convergence, Training Times: Figure 4b shows the log likelihood (LL) of the training data, $P(\boldsymbol{O} \mid \boldsymbol{\Psi})$, versus the iteration number in the EM algorithm (which is used to estimate parameters of *HCFContext*) for different models. We have considered six weeks of training data so, $T = 6 \times 7 \times 24$. We can see that the LLs have converged and the model significantly improves the initial LL values (upto 30%). We can notice that LLs have approximately converged after $n_{con} = 3$ iterations. Even then, the converged LLs have lower values due to large number of values for certain features (such as WiFi APs which has about 440 unique values) which makes the value probabilities, $\phi_{k,f,v}$, very small. In fact, we encountered underflow problem due to multiplication of several small probabilities and then using such a value in the denominator resulting in nan values. In order to solve this problem, we have used scaling approach [27] for α, β values. Figure 4b also shows the time taken in seconds vs. iterations of EM algorithm. The corresponding time for $n_{con} = 3$ is about 200, 800, 1500 seconds respectively for the three models. *These durations are reasonable considering that the training is run offline and very less often.* Both these results are averaged over 4 runs and we can notice that the 95% confidence intervals are too minute to be noticed.

Prediction Performance—A Use Case: We relate to the "lunch" use case mentioned in Sect. I. We considered user A for this purpose and his most related 2 and 3 user groups as found from above—(A, C) and (A, C, D)—and the corresponding models, *HPContext*, *HCFContext* (2), and *HCFContext* (3), respectively (please note this notation to be used in rest of the paper). We first illustrate the prediction

TABLE III
USE CASE ILLUSTRATING THE PREDICTIONS (WITH CORRESPONDING MAX. PROBABILITIES) AT 1 PM ON ONE TUESDAY IN THE TEST PERIOD.

Features	***Ground Truth(A)***	***HPC(A)***	***HCFC(A,C)***	***HCFC(A,C,D)***	***HPC(A) + HCFC(A,C)***	***HPC(A) + HCFC(A,C) + HCFC(A,C,D)***
Wi-Fi AP	00:bl:f2:9b:05:76	**00:og:1f:2e:4n:6c(0.32)**	00:bl:f2:9b:05:76(0.43)	00:bl:f2:9b:05:76(0.41)	00:bl:f2:9b:05:76(0.27)	00:bl:f2:9b:05:76(0.32)
Place Name	F007	F007 (0.84)	**B003 (0.48)**	**J023 (0.31)**	F007 (0.57)	F007 (0.41)
Cell ID	42534164	42534164 (0.62)	42534164 (0.44)	**48759836 (0.3)**	42534164 (0.53)	42534164 (0.39)
LAC	8513	**9353 (0.32)**	8513 (0.47)	8513 (0.57)	**9353 (0.36)**	8513 (0.41)
Battery Level	High	High (0.89)	**Medium (0.41)**	**Medium (0.32)**	High (0.57)	High (0.42)
Battery Status	Discharging	Discharging (0.95)	Discharging (0.93)	Discharging (0.81)	Discharging (0.94)	Discharging (0.89)
Day Period	Afternoon	Afternoon (0.72)	Afternoon (0.82)	Afternoon (0.73)	Afternoon (0.77)	Afternoon (0.75)
Day Name	Tuesday	Tuesday (0.59)	Tuesday (0.68)	Tuesday (0.73)	Tuesday (0.63)	Tuesday (0.67)
Holiday	No	**Yes (0.6)**	No (0.7)	No (0.75)	No (0.55)	No (0.62)

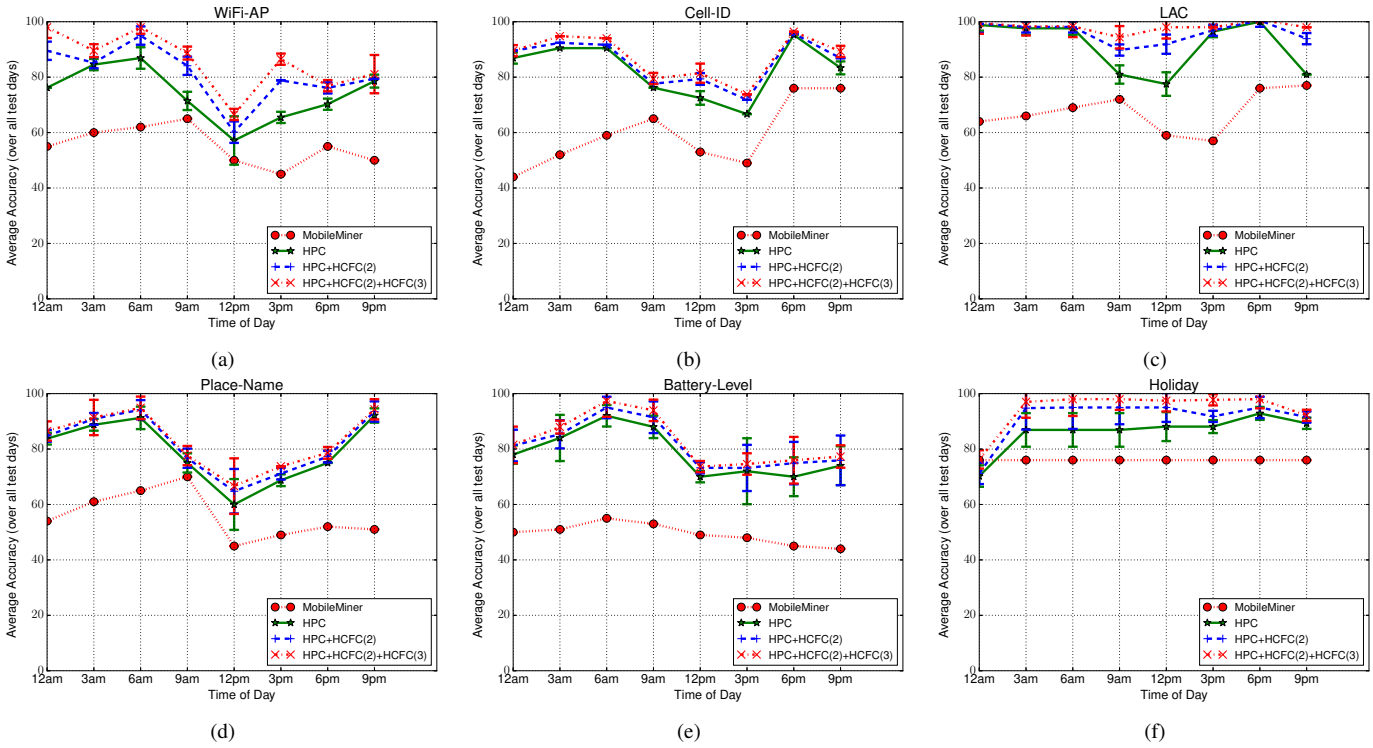


Fig. 5. Average accuracy of the proposed models—*HPContext*, *HPContext* + *HCFContext* (2), *HPContext* + *HCFContext* (2) + *HCFContext* (3) in predicting the respective features at different times in a day (averaged over all 21 test days/4 runs). We can notice a slight drop in accuracy during mid-day (good at other times) owing to a degree of randomness in users motion patterns at those times (still $\approx 75\%$ sufficient for context validation/enhancement).

results using a known use case as follows. We manually observed from the data that users A, C, D usually go to lunch together on weekdays around $12-2$ pm. We wanted to check whether our models are able to capture this group behavior. Hence we predicted the contextual feature-values of user A at 1 pm on one randomly selected weekday (Tuesday) in the test period given the observations of previous one-day duration using the above models. Table III shows the results of different models along with ground truth (column 2). All entries correspond to maximum probability values with those probabilities shown in brackets. Incorrect predictions are depicted in bold. We can notice that *HPContext* (col. 3) does a good job in predicting the personalized features such as *Place Name*, *Battery Level* but makes 3 incorrect predictions for more general features such as *LAC*, *Holiday*, etc. Note that each feature is predicted independently, e.g., predicting *Place Name* correctly does not necessarily mean *Wi-Fi AP* prediction is correct. Interestingly, *HCFContext* (cols. 4,5) makes correct predictions for those general features but fares badly for personalized features. Hence we combined the two models to obtain better predictions (as can be seen in last two columns) as follows. In case of *HPContext* + *HCFContext* (2), we first average the probabilities of values predicted by both models and then take the feature value with the highest average probability; similarly for *HPC* + *HCFC*(2) + *HCFC*(3) (*HPC* refers to *HPContext* and *HCFC* to *HCFContext* for simplicity).

Prediction Performance—Overall: To evaluate the overall performance, we predicted all the contextual feature value pairs of user A at 3 hour increments in the entire 3-week testing period (i.e., $3\times7\times8$ in total) given the past observations of one-day duration. In order to compare with other closest approaches, we considered Mobile Miner [6], which is the current state-of-the-art machine learning algorithm to mine contextual co-occurrences. Each feature is considered to be independently co-occurring with the time of day and day of week (as opposed to sequentially occurring in our case), and is modeled using Multinomial Logistic Regression. Figure 5 shows the average prediction accuracy (percentage of correct predictions) for each feature at different times in a day (averaged over all the 21 days). Even though we are able to predict

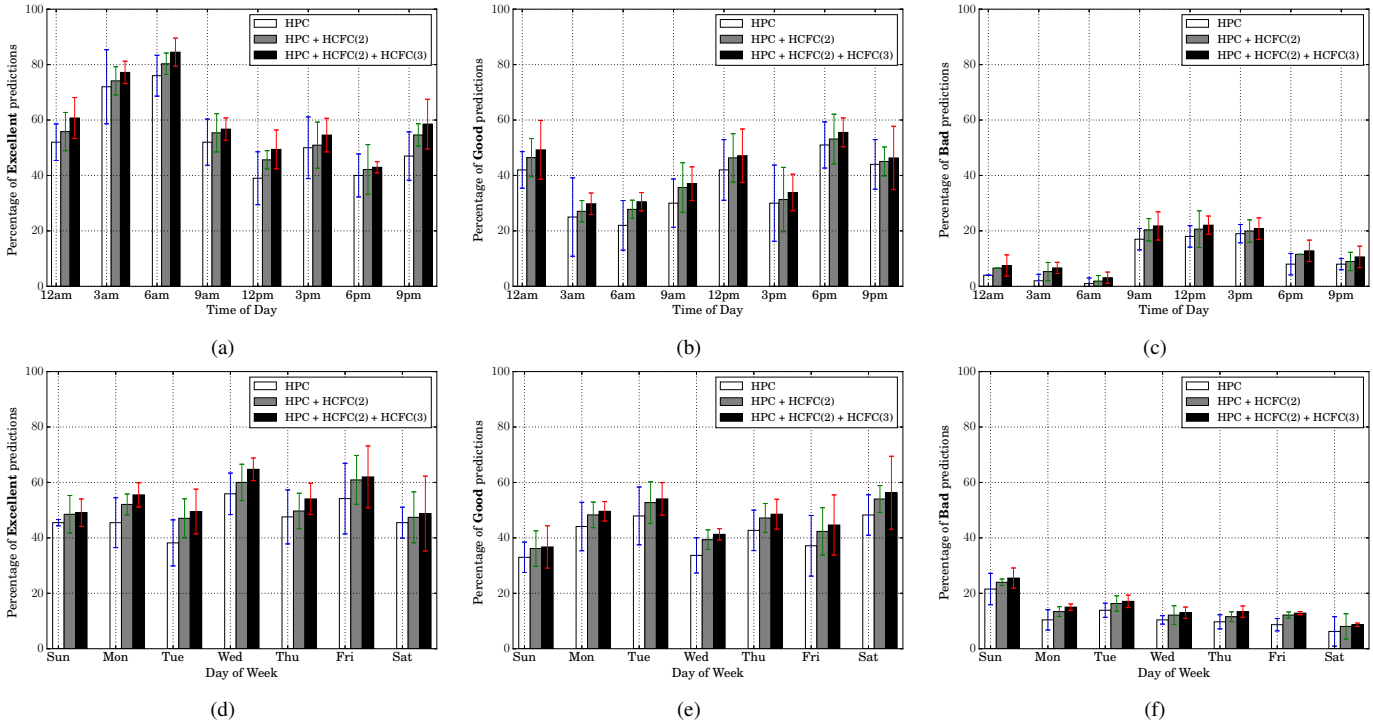


Fig. 6. Percentage of excellent/good/bad predictions—(a)-(c) at different times of day; (d)-(f) for different days of week.

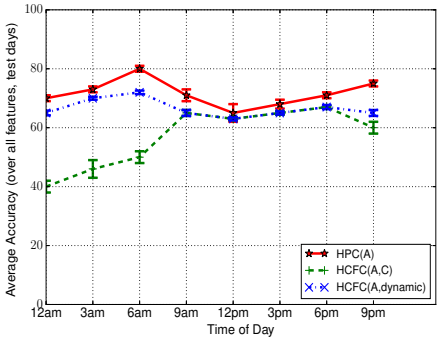
at the hourly level (due to the upsampling mentioned earlier), we show only at 3-hour intervals for clarity. These simulations are also run for 4 runs (to account for randomicities such as random initialization of the model parameters) and the confidence intervals are shown. Due to space limitations, we have shown results only for six features consisting of four location features—Wi-Fi AP, Place Name, Cell ID, Location Area Code (LAC), one device feature—Battery Level and one time feature—Holiday. Others follow similar pattern.

First of all we can notice that the proposed models perform better than Mobile Miner. Second, in case of proposed models, we can notice that the accuracy slightly drops during mid-day compared to other times. The reasons for this drop as follows—(1) the user is more mobile during those times introducing randomness into the data making the model hard to learn; (2) the average number of places visited by each user is 270 (as indicated above) and the K value chosen (as a tradeoff between complexity and accuracy) is less than that; (3) the data is down-sampled to one hour interval, meaning, any places with stay duration less than one hour will not be captured well. However it is important to note that the accuracy in such cases is still on an average about 75% which is reasonable to validate/complement the context from sensors. During other times of the day, we can notice that the accuracy is about 90%, which means the models are able to predict the values of those contextual features correctly 90% of the time. Third, we can notice that *HPC* + *HCFC*(2) model improves accuracy over *HPC* to a maximum of 15% especially in more general features such as *LAC*, *Holiday*. In case of personalized features such as *Place Name*, *Battery Level*, the improvement is minor. We also notice that the additional improvement from *HCFC*(3) is again minor, about 5%. In total, *HCFC* contributes upto 20% improvement in accuracy. We have also plotted similar results for each day of the week but could not include them due to space limitations. In addition to above trends in generic/personalized features vs. models, we have noticed a drop in accuracy over weekends (to $\approx 75\%$ on average) again owing to increased mobility with less sequential behavior. The average accuracy is about 80% to 90% for the rest of the days (*HPC* + *HCFC*(2) + *HCFC*(3) model).

Prediction Quality: Next, we tested our models' prediction quality by testing how many of the features (all in Table II) the models are able to predict correctly at a given time of day. For this purpose, we created three categories—*Excellent*, *Good*, *Bad*. If the model is able to predict at least 7 features out of 9 correctly at a given instant, we call it *Excellent* prediction. Similarly we call 4/5/6 features prediction, a *Good* prediction and 1/2/3 (0 is not included) features prediction, a *Bad* prediction. For example, the prediction corresponding to $HPC(A)$ in Table III is considered a *Good* prediction, while that belonging to $HPC(A)+HCFC(A,C)$, an *Excellent* prediction. Figure 6 shows these results (averaged) for different times of day and days of week. In both sets of figures, we can notice that the percentage of *Excellent* cases is at least 50%. Secondly, the percentage of *Excellent* cases is more than *Good* cases which in turn is more than the *Bad* cases (in particular the *Bad* cases are very less comparatively).

Test Data Rotation: The performance of the models when the test data chosen is the first (1-3), middle (4-6) and last (7-9) three weeks is shown in Fig. 4c, which shows the percentage of correct predictions across all features and test days for different choices of test data (results shown only for *HPC* + *HCFC*(2) + *HCFC*(3) for clarity). We can notice that the performance is roughly the same showing robustness of the proposed models to choice of test data and their ability to fully learn users' sequential patterns using 6 weeks train data.

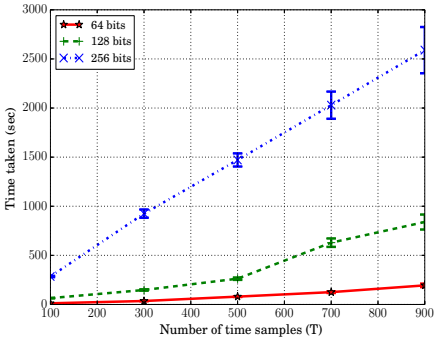
Contextual Optimal User Group Selection: So far, for a given user, A, we have found the optimal user groups considering all times of the day. However, it is more beneficial to find



(a)

n (bits)	c	$P(O \mid \Psi)$	π	ρ	θ
64	10^3	5.70	6.56	9.10	14.95
128	10^3	5.10	4.76	8.26	11.84
256	10^3	4.01	3.48	6.89	10.02
512	10^3	1.83	2.95	6.05	7.23
64	10^6	0.15	1.21	4.04	7.12
128	10^6	0.13	1.05	3.09	6.04
256	10^6	0.11	0.58	1.98	5.14
512	10^6	0.11	0.45	0.97	2.50

(b)



(c)

Fig. 7. (a) Comparison of performance among *HPContext*, *HCFContext* (with fixed optimal user group), *HCFContext* (with dynamic optimal user group). (b) Worst-case errors of parameters [10 runs] (%). (c) Time taken to run Algorithm 1 for different number of training (time) samples vs. key length (bits).

the optimal user group based on the time of the day as is the real scenario. For example, for a given user, the closely related users during the office hours may be different from the closely related users during home hours. Taking this idea into account, we have found the most closely related user for user A (i.e., optimal 2-user group) at different times of the day using the perplexity method mentioned earlier. Instead of averaging across all times of the day, we find the user group with minimum average perplexity at each time instant of the day. The results are as follows—$(12am, B)$, $(3am, B)$, $(6am, B)$, $(9am, C)$, $(12pm, C)$, $(3pm, C)$, $(6pm, C)$, $(9pm, D)$.

Prediction Using Only *HCFContext*: We now predict the feature-value pairs of user A at all time instants in the test period using only the collaborative filtering model, *HCFC*(2), with dynamic optimal 2-user group obtained from above. Figure 7a shows the performance of that model compared against two other models—the personalized model *HPC* and *HCFC*(2) with fixed optimal 2-user group (A, C). We can notice that dynamic *HCFC* performs close to *HPC* and better than fixed *HCFC*. This indicates that personalized context can be obtained from collaborative filtering of contexts corresponding to user's closely related users with appropriate dynamic (i.e., time of day/activity the user is performing) selection of closely related users.

Privacy-preserving Algorithms: To evaluate the performance of Algorithms 1, 2, we tested them on a simple HMM with $M = 2$ parties, $K = 2$ hidden states and $|\boldsymbol{f}_{tu} = 6|$ observation states per hidden state. We evaluated both the amount of error introduced (due to scaling as mentioned in Sect. III-B)) as well as the time taken to train the HMM and run the predictions. For the former, we calculated the error by comparing with the non-privacy preserving case. We varied the key length (bits) n and the scaling factor c. The worst-case errors over 10 runs with $T = 1000$ samples, as percentages, for different parameters is shown in Fig. 7b. We notice that the error reduces as scaling factor increases (as expected). Similarly, as the key length increases, the error reduces and also security increases. We can notice that for large keys and reasonable scaling factors, the error due to integer approximation and consequent over- or underflow is insignificant (about 2% in the worst case over all parameters). This shows that the effect on the prediction accuracy results above will not be drastic. However, the price is in terms of run-time. Figure. 7c shows the average run-times of Algorithm 1 vs. the number of time samples as well as the key length (with $c = 10^6$). We can see that the time taken varies linearly with the number of samples used. However, the relation seems to be approximately quadratic with key-length. This result shows the tradeoff between security and run-time. As the key-length is increased, more is the security, less is the amount of error introduced, however the run-times are more. Hence a suitable key length should be chosen that is a compromise between security/error and run-time. The run-times for Algorithm 2 are on average five times more. *However these algorithms need to be run only for training the HCFContext which is run very less frequently and offline.* Moreover, we will leverage recent advances in encryption and multi-party computation algorithms such as [28] to help further reduce the runtimes of these algorithms, as part of future work.

Real-time Inference and Cold-start: Once the training is complete, model parameters are known to each device. Prediction, then, just involves evaluating (12), (13) by plugging in the learned parameters. These are just a few arithmetic operations and do not involve any compute-intensive encryption/decryption algorithms unlike training which happens offline. *Hence, our approach will not have any problems in practical implementations i.e., making predictions in real time.* Furthermore, to combat the cold-start problem akin to collaborative filtering based approaches, we suggest to—(i) use sensor context and also obtain user validation for additional security; (ii) use sensor context + *HPContext* until *HCFContext* is learnt well.

Energy Considerations: Note that all of the features we worked with are passive, i.e., do not require active probing that consumes energy. One exception is the Wi-Fi, which needs to be turned ON in case it is not ON. In all other cases, we piggyback on the sensor data already available on the phone, reducing energy consumption.

V. Conclusion and Future Work

We proposed and evaluated (on a real-life dataset with over 80% accuracy) privacy-preserving, sequential history-based personalized and collaborative-filtering models, for current and future mobile context prediction to validate and/or enhance the sensor context. Their feasibility for practical deployment in security applications and/or mobile personal assistant technologies is shown. As future work, we plan to conduct a pilot study of our models; improve their training times leveraging suboptimal algorithms and enhance their prediction accuracy.

Acknowledgment

We thank the US Department of Homeland Security Science & Technology Directorate (DHS S&T) Cyber Security Division for their support under the contract No. D15PC00159.

REFERENCES

[1] Niantic, "Pokemon Go," http://www.pokemongo.com, 2016.

[2] V. Pejovic and M. Musolesi, "Anticipatory Mobile Computing: A Survey of the State of the Art and Research Challenges," *ACM Comput. Surv.*, vol. 47, no. 3, 4 2015.

[3] G. Salles-Loustau, L. Garcia, K. Joshi, and S. Zonouz, "Don't just BYOD, Bring-Your-Own-App Too! Protection via Virtual Micro Security Perimeters," in *IEEE/IFIP International Conference on Dependable Systems Networks*, 6 2016.

[4] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful GPS spoofing attacks," in *Proceedings of the 18th ACM conference on Computer and communications security - CCS '11*. New York, NY, USA: ACM Press, 2011, p. 75.

[5] T. Bao, H. Cao, E. Chen, J. Tian, and H. Xiong, "An Unsupervised Approach to Modeling Personalized Contexts of Mobile Users," in *2010 IEEE International Conference on Data Mining*. IEEE, 12 2010, pp. 38–47.

[6] V. Srinivasan, S. Moghaddam, A. Mukherji, K. K. Rachuri, C. Xu, and E. M. Tapia, "MobileMiner: mining your frequent patterns on your phone," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '14 Adjunct*. New York, NY, USA: ACM Press, 2014, pp. 389–400.

[7] A. Mukherji, V. Srinivasan, and E. Welbourne, "Adding intelligence to your mobile device via on-device sequential pattern mining," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct Publication - UbiComp '14 Adjunct*. New York, NY, USA: ACM Press, 2014, pp. 1005–1014.

[8] K. Farrahi and D. Gatica-Perez, "A probabilistic approach to mining mobile phone data sequences," *Personal and Ubiquitous Computing*, vol. 18, no. 1, pp. 223–238, 1 2014.

[9] A. Mannini and A. M. Sabatini, "Machine Learning Methods for Classifying Human Physical Activity from On-Body Accelerometers," *Sensors*, vol. 10, no. 2, pp. 1154–1175, 2 2010.

[10] D. K. Jonathan Feng-shun Lin, "Automatic Human Motion Segmentation and Identification using Feature Guided HMM for Physical Rehabilitation Exercises," in *IEEE/RSJ Int. Workshop Conf. Intelligent Robots and Systems (IROS), Robot. Neurology Rehab.*, 2011.

[11] L. A. Castro, J. Beltrán, M. Perez, E. Quintana, J. Favela, E. Chávez, M. Rodriguez, and R. Navarro, "Collaborative Opportunistic Sensing with Mobile Phones," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, ser. UbiComp '14 Adjunct. New York, NY, USA: ACM, 2014, pp. 1265–1272.

[12] J. Mantyjarvi, J. Himberg, and P. Huuskonen, "Collaborative context recognition for handheld devices," in *Proc. of the International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 3 2003.

[13] E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell, "Darwin Phones: The Evolution of Sensing and Inference on Mobile Phones," in *Proc. of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*. New York, NY, USA: ACM, 2010.

[14] H. Polat and W. Du, "Privacy-preserving collaborative filtering using randomized perturbation techniques," in *Proceedings of the Third IEEE International Conference on Data Mining*. Melbourne, Florida: IEEE Computer Society, 2003, p. 756.

[15] R. Parameswaran and D. M. Blough, "Privacy Preserving Collaborative Filtering Using Data Obfuscation," in *2007 IEEE International Conference on Granular Computing (GRC 2007)*. IEEE, 11 2007, pp. 380–380.

[16] S. Guo, S. Zhong, and A. Zhang, "A Privacy Preserving Markov Model for Sequence Classification," in *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics - BCB'13*. New York, NY, USA: ACM Press, 2007, pp. 561–568.

[17] S. Renckes, H. Polat, and Y. Oysal, "Providing predictions on distributed HMMs with privacy," *Artificial Intelligence Review*, vol. 28, no. 4, pp. 343–362, 12 2007.

[18] H. Kikuchi, H. Kizawa, and M. Tada, "Privacy-Preserving Collaborative Filtering Schemes," in *2009 International Conference on Availability, Reliability and Security*. IEEE, 2009, pp. 911–916.

[19] H. X. Nguyen and M. Roughan, "Multi-Observer Privacy-Preserving Hidden Markov Models," *IEEE Transactions on Signal Processing*, vol. 61, no. 23, pp. 6010–6019, 12 2013.

[20] Wikipedia, "Plate Notation," https://en.wikipedia.org/wiki/Plate_notation.

[21] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[22] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Advances in Cryptology — EUROCRYPT '99*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238.

[23] M. Pathak, S. Rane, W. Sun, and B. Raj, "Privacy preserving probabilistic inference with Hidden Markov Models," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5 2011, pp. 5868–5871.

[24] A. Chopra, "Comparative Analysis of Key Exchange Algorithms in Cryptography and its Implementation," *IMS Manthan (The Journal of Innovations)*, vol. 8, no. 2, 2015.

[25] Y. Chon, E. Talipov, H. Shin, and H. Cha, "Mobility prediction-based smartphone energy optimization for everyday location monitoring," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems - SenSys '11*. New York, NY, USA: ACM Press, 2011, p. 82.

[26] G. Heinrich, "Parameter estimation for text analysis," University of Leipzig, Tech. Rep., 2008. [Online]. Available: https://faculty.cs.byu.edu/~ringger/CS679/papers/Heinrich-GibbsLDA.pdf

[27] P. Blunsom, "Hidden Markov Models," Dept. of Computer Science, Utah State University, Tech. Rep., 2004. [Online]. Available: http://digital.cs.usu.edu/~cyan/CS7960/hmm-tutorial.pdf

[28] R. Bitar, P. Parag, and S. El Rouayheb, "Minimizing Latency for Secure Distributed Computing," in *IEEE International Symposium on Information Theory (ISIT)*. Aachen: IEEE, 2017.