Pervasive and Mobile Computing 12 (2014) 232–243

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

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

Fast track article

Adaptive and context-aware privacy preservation exploiting user interactions in smart environments

Gautham Pallapa [a,*], Sajal K. Das [b], Mario Di Francesco [c,d], Tuomas Aura [c]

[a] West Interactive Corporation, United States

[b] Computer Science Department, Missouri University of Science and Technology, United States

[c] Department of Computer Science and Engineering, Aalto University, Finland

[d] Department of Computer Science and Engineering, University of Texas at Arlington, United States

ARTICLE INFO

Article history:
Available online 27 December 2013

Keywords:
Pervasive systems
Context awareness
Privacy preservation
User interactions
Smart environments

ABSTRACT

In a pervasive system, users have very dynamic and rich interactions with the environment and its elements, including other users. To efficiently support users in such environments, a high-level representation of the system, called the context, is usually exploited. However, since pervasive environments are inherently people-centric, context might consist of sensitive information. As a consequence, privacy concerns arise, especially in terms of how to control information disclosure to other users and third parties. In this article, we propose context-aware approaches to privacy preservation in wireless and mobile pervasive environments. Specifically, we design two schemes: (i) to reduce the number of interactions between the user and the system; and (ii) to exploit the interactions between different users. Both solutions are adaptive and, thus, suitable for dynamic scenarios. In addition, our schemes require limited computational and storage resources. As a consequence, they can be easily implemented on resource-constrained personal communication and sensing devices. We apply our solutions to a smart workplace scenario and show that our schemes protect user privacy while significantly reducing the interactions with the system, thus improving the user experience.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Technological advances and the widespread adoption of both sensing and personal communication devices, such as smartphones and PDAs, are making pervasive systems and the Internet-of-things a reality [1]. This large number of heterogeneous devices not only makes possible to access information "everywhere, anytime" but also, more significantly, supports the daily activities and occupations of people seamlessly [2]. Actually, pervasive systems would not exist without people. Conversely, people would not use systems which are obtrusive, difficult to use, or subject to risks not proportionate to their benefits [3]. Hence, it is extremely important to consider user interactions in the design of pervasive systems [4].

Smart environments represent a specific type of pervasive systems in which the people's surroundings are enriched with sensing and actuation capabilities, with the purpose of improving the user experience [5]. One fundamental aspect of such systems is intelligence, in the sense that they can autonomously acquire and apply knowledge about the environment and its actors [6]. Indeed, intelligence is needed due to a number of factors. First, the interactions between people and different devices need to be abstracted and interpreted to obtain meaningful decisions and actions. In fact, the amount of raw data could be enormous, and the related patterns complicated. Second, the interactions between people and smart environments

* Corresponding author.
E-mail address: gpallapa@west.com (G. Pallapa).

1574-1192/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.pmcj.2013.12.004

should be so natural that they cannot even be noticed [7]. However, these interactions, which also involve different sources of information, are usually very dynamic and complex. Hence, significant effort might be required to take autonomous decisions, as well as to adapt to individual users as they interact with the environment [8].

One of the foundations of pervasive systems and smart environments is the so called *context*, defined as the circumstances or background elements which (help to) determine the meaning of an event or the occurrence of specific patterns. Context-awareness is also the cornerstone of pervasive applications [9]. In most cases, context information is people-centric, hence it may consist of sensitive information which may lead to security and privacy issues. In this article, we focus on *privacy*, defined as the claim of an individual to allow or deny, according to certain constraints, disclosure of personal information to third parties [10]. Note that third parties are not only other people, but also entities (e.g., applications or providers) which could potentially exploit context information to provide services. This makes the problem even more challenging [5, Chapter 1].

A possible solution consists of an explicit mapping between the information available in the smart environment and a numerical privacy value. In this approach, the privacy value denotes how much the user is willing to share the related information with other actors in the system [11]. However, this solution alone is rather limited, since it is not enough flexible to characterize complex and dynamic scenarios, or even to tackle interactions between different users. In addition, it may require many explicit interactions with the user. As a consequence, the system might be not very user-friendly, or even inconvenient if the user intervention is requested too often.

In this article, we propose two context-aware approaches to privacy preservation in wireless and mobile pervasive environments. Specifically, we introduce a hybrid scheme to reduce the interactions between the user and the system. We also present an approach based on user behavior which is capable to exploit the interactions between different users and the environment. Both of our solutions are adaptive, thus suitable for dynamic scenarios. In addition, our schemes require limited computational and storage resources, so that they can be implemented on resource-constrained personal communication and sensing devices. We apply our solutions to a smart workplace scenario, and show that our schemes are effective in protecting user privacy. Furthermore, we show by simulation that our proposed approaches also significantly improve the user experience, in terms of reduced number of explicit interactions with the system.

The rest of the article is organized as follows. Section 2 overviews the related work, while Section 3 introduces the reference system architecture and the considered privacy model. Section 4 proposes two schemes for privacy preservation in smart environments. Section 5 discusses an application of the proposed schemes in the specific scenario represented by smart workplaces. Section 6 presents the simulation results, while Section 7 concludes the article.

2. Related work

Security issues have been extensively studied in the context of both pervasive computing [12] and smart environments [5, Chapter 11]. However, only a few approaches specifically targeted privacy preservation with focus on user experience.

The study in [13] showed how privacy is perceived by users in a smart environment. However, the findings are not specifically targeted on context, but rather focused on how users are aware of privacy issues related to a specific component (i.e., a sensing or tracking device). The work in [14] explored the risks associated with personal privacy in pervasive scenarios, and provided a comprehensive set of guidelines on how to trade off privacy preservation and user benefits in such systems. However, it did not specifically investigate inter-related and contextual information.

Many works focused on the problem of location privacy [15]. Even though location can provide context information, actually it is only a specific attribute of context. Hence, these solutions cannot generally be applied to other context elements, nor can they be extended to scenarios where multiple elements are considered. Many other solutions were also aimed at providing anonymity [16]. A related privacy model that has emerged in the last few years is given by differential privacy [17]. Specifically, the major objective of the latter is to anonymize large datasets so that adversaries cannot relate released data to specific individuals. Indeed, anonymizing data does not apply to the case where a user wants to share information, but needs to control how to release it in order to protect their privacy.

Among the solutions that addressed privacy concerns with specific reference to context information, the authors in [12] investigated the relation between privacy and context. Although they proposed a framework for modeling privacy in context-aware systems, they did not consider user interactions. A scheme to detect and resolve potential privacy disclosure in pervasive systems was introduced in [18]. The proposed approach is based on an access-rights graph, which is exploited to make decisions on how specific context information can be accessed. However, the focus of the article is more on access control, rather than on privacy preservation. An architecture for privacy preservation in opportunistic context-aware systems was presented in [19]. Even though the system leverages user interactions, the proposed solution is still based on anonymity.

Privacy preservation was considered in [11], which explicitly addressed how to quantify the privacy associated with the different elements in a pervasive system. The proposed approach is based on different privacy levels, which are assigned to the distinct elements in a context. This basic information is then abstracted into context and privacy states, and modeled through a Context-Privacy Graph (CPG). Two different schemes for privacy preservation are then introduced: a user-centric approach, where a global allowed privacy level is statically assigned by the user; and a system-centric approach, where the privacy levels are automatically assigned by the system. Since the considered schemes are radically different, they

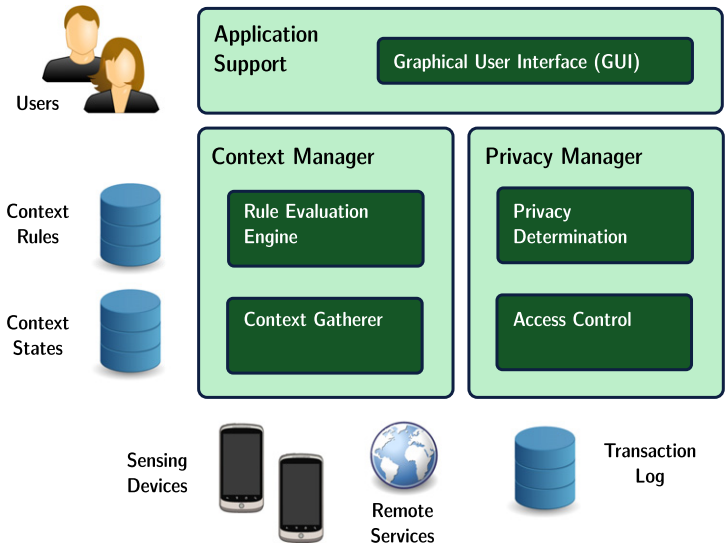

Fig. 1. Components of a privacy and context-aware pervasive system.

do not provide enough flexibility to adapt to users' needs. In addition, the system-centric approach requires considerable computational resources, thus not being suitable to constrained pervasive devices.

A privacy-preserving architecture for smartphone applications was introduced in [20]. The proposed solution specifically addresses interest sharing and privacy scheduling, in terms of how to disclose availability and location while preserving privacy. Similar to our approach, the one in [20] exploits a threshold-based mechanism to perform decisions about sharing. However, that solution relies on a semi-trusted server, while here we opportunistically derive sharing of information based on co-located nodes without the need for external entities. A privacy framework for context-aware applications was developed in [21]. The authors present a few case studies as well as a prototype implementation for Android platforms. However, the proposed approach has not been quantitatively evaluated. In contrast, in this work we also evaluate the performance of our proposed privacy preservation schemes.

Different from most of the approaches already proposed in the literature, such as [15], we do not limit to location privacy only, but address generic context information. In addition, we do not provide anonymization as in [16,19], but rather quantify the privacy associated with context information, similarly to [11]. In contrast with the latter, however, we exploit a hybrid approach which trades off the radically different options of "no configuration" and "static user configuration". To a certain extent, our solution is similar to [18], even though it is not focused on access control. However, different from both [18,11], we do not rely on a graph structure, and derive the privacy of context elements more naturally from user interactions.

3. Preliminaries

In this section, we will first present the fundamental concepts behind our proposed adaptive privacy preservation schemes, then we will introduce the system architecture and its major components.

3.1. System model and reference architecture

We denote a pervasive environment E consisting of a set of n resources as $E = \{r_1, r_2, \ldots, r_n\}$. Each resource corresponds to either a sensing unit (e.g., a sensor measuring temperature or a camera) or an attribute made available by a (remote) service (e.g., weather forecast or traffic information). Resources are characterized by means of *context elements* e_i for $i = 1, \ldots, n$, each containing the basic information needed by the system: a resource ID, the associated data, and a timestamp. We define a context as a collection of context elements. To avoid ambiguity, and since the state of the system can be derived depending on the context information, we will refer to a specific context as *context state*. More precisely, a context state C_j is defined as a set of m context elements such that $C_j = \{e_1, e_2, \ldots, e_m\}$, where $m \leq n$. Context states are processed according to user or system-defined *context rules*.

These fundamental elements are managed by the system through the components shown in Fig. 1. Specifically, a *context manager* is used to handle context information. The context manager is made of two different subsystems: (i) a *context gatherer* which retrieves the resources, builds the context elements, and associates them to context states; and (ii) a *rule engine* which evaluates the context rules and updates the state of the system accordingly. In order to keep track of the current state, the system stores both the context rules and the context states in databases, which are made available to the different components. Privacy-preservation is accomplished by means of a *privacy manager*, which associates privacy information

with the different context states and ensures proper access to sensitive information. The privacy manager maintains a transaction log, which is used to update the privacy of the different context states according to specific policies. The *application support* component contains the different user applications which can take advantage of the privacy-preservation service. In addition, it comprises interfaces for configuring the different components of the system, including the privacy settings and the context rules.

In the following subsection, we will introduce the privacy model used in the rest of the article. We will assume that the context manager is already available – e.g., by using the solution in [22] – and focus on the privacy manager instead.

3.2. Privacy representation and management

In our scheme, we associate a *privacy value* with each element in the system, representing the related amount of privacy [11]. To keep the user interface simple, users are asked to assign privacy values from a small set L of keywords or numbers, such as the scale 1–5. These labels are mapped to the actual numerical privacy values $0 \leq w(e_i) \leq 1$ for all context elements e_i. The privacy value of a set of context elements such as $C_j = \{e_1, e_2, \ldots, e_m\}$ is defined as the sum $w(C_j) = \sum_i w(e_i)$. In addition to those individual privacy values, a *global privacy setting* π is also defined. The global privacy setting summarizes how context information is to be shared or protected by the user.

Privacy management is performed as follows. First, privacy values for context elements and context states are determined. Second, those values are evaluated with respect to the global privacy setting each time context information is requested. If the current global privacy setting allows the disclosure of the context information, then the related context elements are shared with the requesting party. Otherwise, they are denied, and eventually the user is prompted for additional actions, e.g., to allow an exception or to change the current privacy settings so that such a request can be satisfied.

4. Adaptive privacy preservation

In a pervasive environment, the interactions between users and resources are inherently rich and evolve rapidly with time. Therefore, an effective privacy preservation scheme should track the context elements and update the status of the system accordingly. Hence, the privacy preservation scheme should be able to dynamically adapt to the pervasive environment as soon as some of the system elements change.

In the previous section, we have (implicitly) characterized the status of the system at a specific instant. The resulting characterization can also be thought as a snapshot of the system at a given time. In order to track the dynamics of the system, we extend the basic model to time-varying processes by introducing the time variable t. As a result, we will denote as $w(e_i, t)$, $w(C_j, t)$, and $\pi(t)$ the context elements, context states, and the global privacy settings of the system at time t, respectively.

We define the state of the system at a given time t as a *session*, denoted as $S(t)$. If context rules do not change with time, the state of the system is univocally identified by the set of context elements and context states, along with the corresponding privacy levels and the global privacy setting. If the privacy value of context states can also be derived according to a certain policy, then the state of the system is univocally determined just by the context elements and their weights, which we denote as $S(t) = \{w(e_1, t), w(e_2, t), \ldots, w(e_m, t)\}$. In the following, we will assume that the context rules are statically defined, so that we can characterize a session only through the weighted context elements.

4.1. A hybrid approach to privacy preservation

In this section, we will present a hybrid approach to privacy preservation. The proposed scheme exploits both user interactions and automatic determination of privacy values associated with context states, based on the context elements.

Let us consider, without loss of generality, a scenario where two[1] users u_s and u_r interact within a session. Specifically, user u_s is *sharing* some information according to some privacy settings, and user u_r is *requesting* some information, described in terms of the set of context elements $C_r(t) = \{e_1(t), e_2(t), \ldots, e_r(t)\}$. We now describe how our scheme, executed at the user u_s, results in sharing the relevant context information with the user u_r while preserving the desired level of privacy.

We assume that a global privacy setting $\pi(t)$ is available, for instance as specified by the user u_s. We also assume that the privacy values (i.e., weights) of the context elements $w(e_i, t)$ are available. In order to determine if the request can be satisfied, the system compares the privacy of the context states to the global privacy setting, i.e.,

$$w(C_r, t) = \sum_{i=1}^{r} w(e_i, t) \leq \pi(t) \tag{1}$$

where r is the number of context elements in the request $C_r(t)$. If Expression (1) is satisfied, then the user u_s is notified of the corresponding session and of the information shared. Otherwise, the user is prompted for an action, i.e., to allow or deny the disclosure of the related context information.

[1] Session involving multiple users can be referred to as a set of pairwise sessions between one user sharing information and another user requesting a context.

Algorithm 1: Scheme based on a hybrid approach

input: request $C_r(t) = \{e_1(t), e_2(t), \ldots, e_r(t)\}$
if $\sum_{i=1}^{r} w(e_i, t) \cdot x_i \leq \pi(t)$ **then**
  inform user of the session $C_r(t)$;
else
  build the decision tree;
  obtain $C_s(t)$;
  inform user of the session $C_s(t)$;
  query permissions for elements of $C_r(t) \setminus C_s(t)$;

Our goal here is to keep the system as autonomous as possible, thus reducing the number of interactions with the user to the minimum extent. To this end, we try to maximize the number of context elements which can satisfy the request, in terms of the constraint in Expression (1). More formally, we can map the considered approach to a 0–1 knapsack problem [23], which can be formulated as follows:

$$\max \sum_{i=1}^{r} w(e_i, t) \cdot x_i \tag{2}$$
$$\sum_{i=1}^{r} w(e_i, t) \cdot x_i \leq \pi(t)$$
$$x_i \in \{0, 1\}$$

where x_i is a binary value and denotes if the context element $e_i(t)$ is included in the context information which can be automatically shared, according to the current global privacy setting. The 0–1 knapsack problem can be solved through linear programming [24]. However, to reduce the computational requirements of our scheme, we used a greedy approach based on a decision tree (more details will be provided in Section 5.2).

Let $C_s(t)$ be the set of shareable context elements resulting from Expression (2). Then, the user u_s is informed of the corresponding session $C_s(t)$, and asked for permission to share the remaining context elements which do not belong to $C_s(t)$, i.e., the elements in $C_r(t) \setminus C_s(t)$.

The hybrid approach is illustrated by Algorithm 1.

The proposed approach has a number of advantages. In fact, if Expression (1) is satisfied, the user u_s is just informed of the session, and no further action is needed. Otherwise, the user u_s has to approve the sharing of a limited number of context elements. This approach overall reduces the number of both queries and user prompts, thus allowing a smoother interaction with the pervasive environment. In addition, the user experience is improved since the system allows the user to know at a glance the context elements that have been shared, as well as to modify the privacy levels of the context elements.

The linearity of the model is clearly a simplification. In some situations, the context elements could be dependent on each other, and the cumulative privacy value of their disclosure could be higher or lower than the sum of the individual elements. The model presented here can actually be extended to cover the non-linear cases; however, at the cost of a more complicated user interface and higher computational demands. We believe that, in order to be adopted by actual users, the scheme should be as easy to understand as possible, even if that imposes some limitations on its expressivity. Typically, such user-centered schemes evolve over time and are extended to support the use cases that are most demanded by the users. Without feedback from initial deployments and pilots, it is not possible to determine which non-linear use case will actually be demanded by users.

4.2. Privacy preservation based on user behavior

So far we have characterized privacy management by referring to a session which occurs between two users. However, interactions within a pervasive environment usually have recurring patterns, mainly related to locations, activities, and also social interactions between users. As a consequence, privacy preservation can be further enhanced by explicitly considering these aspects in the design of privacy management schemes. Specifically, the user behavior can be embedded in the privacy preservation scheme.

The main idea behind our approach is to exploit the history of interactions between users and the environment in order to obtain the amount of privacy associated with specific requests. Now, the history can be built on the basis of the individual context requests, so that there is a direct mapping between the requesting user and the queried context state. However, this would require storing a large amount of data because a single request can consist of any subset of the context elements as they are defined in the system. In order to reduce both the implementation complexity and the storage requirements, we associate the history information with individual context elements, so that the corresponding solution is suitable to resource-constrained personal and sensing devices, and then extend it to context requests.

Algorithm 2: Scheme based on user behavior

```
foreach element e_i in C_r from u_k do
    Pr(e_i(t)|u_k) ← n(e_i(t),u_k) / Σ_{j=1}^{N} n(u_j);
    if Pr(e_i(t) | u_k) ≥ (1/r) Σ_{j=1}^{s} Pr(e_j(t) | u_k) then
        add e_i to C_s;
inform user of session C_s;
if C_r(t) ≠ C_s(t) then
    query permissions for elements in C_r(t) \ C_s(t);
    update C_s with the new permissions;
foreach element e_i in C_s do
    if e_i has been shared then
        n(e_i(t), u_k) ← n(e_i(t), u_k) + 1;
n(u_k) ← n(u_k) + 1;
```

Before proceeding further, let us recall that we are considering the scenario where a user u_s is sharing context information with others according to certain global privacy settings. We also assume that the user u_s receives a request of context information in the form $C_r(t) = \{e_1(t), e_2(t), \ldots, e_r(t)\}$ by some other user u_k. Let us also define as $n(e_i(t), u_k)$ the number of times the context element $e_i(t)$ has been shared between u_s and u_k, and $n(u_k)$ as the total number of sessions between u_s and u_k. Thus, we can estimate the probability that the context element c_i is shared with user u_k as:

$$\Pr(e_i(t) \mid u_k) = \frac{n(e_i(t), u_k)}{\sum_{j=1}^{N} n(u_j)},$$

where N is the total number of users which have requested context information for (i.e., which have already had sessions with) u_s in the past. A context element e_i in C_r is shared if the corresponding probability is above the average sharing probability for all elements in the request, i.e.,

$$\Pr(e_i(t) \mid u_k) \geq \frac{1}{r} \sum_{j=1}^{r} \Pr(e_j(t) \mid u_k), \tag{3}$$

where r is the number of context elements in the request.

Expression (3) is evaluated for all context elements in the request. Those which satisfy the constraint are added to the set of shared context elements $C_s(t)$. If all context elements can be disclosed, i.e., $C_s(t) \equiv C_r(t)$, then they are automatically shared and the user is just informed of the session. Otherwise, the user is prompted to explicitly allow or disallow the sharing of the context elements in $C_s(t) \setminus C_r(t)$, as already discussed in Section 4.1.

The approach based on user behavior is illustrated by Algorithm 2.

5. Case study: privacy preservation in smart workplaces

Smart workplaces are among the most significant examples of smart environments [5, Chapter 14]. In this context, several solutions focus on how to instrument and build an intelligent workspace [25], mostly in the context of office environments [26]. The importance of user interactions and behavior in this kind of smart environments has been clearly pointed out in [27].

In the following, we will present an instance of a smart workplace scenario. It is not related to a specific workspace application and consists of two different use cases in which user interactions are explicitly considered. This scenario will then be used to exemplify how the proposed privacy preservation approach works in a real setting.

5.1. Reference scenarios

The specific smart workplace scenario we consider can generally be applied to all cases where an employee (or worker): does not require access to special manufacturing equipment or tools; and interacts directly with customers. As a consequence, it is particularly suitable for employees of companies which provide services to their customers. We introduce two concrete use cases as follows.

- *Telework.* Angela usually shares her phone number and address with both co-workers and customers in order to assist them in their business. Angela spends part of the time in her company's office; however, she teleworks from home for a few days per week. In such a situation, Angela would like to hide her personal information including her location, phone

number and address. In fact, Angela does not intend to disclose where she lives, nor does she want to receive phone calls at home, as she usually works late when her children are sleeping.

- *Customer loyalty*. Angela's company compiles market forecasts for individual investors and financial planners. Those forecasts are updated periodically, and only the complete and thoroughly reviewed releases are made available to customers. To better track the market dynamics, the employees have work-in-progress documents, which are generally not disclosed to customers. However, those internal documents are still valuable, and can be made available to loyal customers at discretion of the employee.

Note that both use cases involve interactions between users of different classes, such as co-workers and customers. The first use case involves location information, while the second one is mainly related to trust between users. Even though these use cases might appear quite specific, they can easily be adapted to more general scenarios as well.

5.2. Analysis

In this section, we will consider the above scenario and illustrate how our scheme can be effectively exploited for the different use cases.

For the *telework* use case, let us assume that the context elements in the system include the user's phone number and address,[2] location, and activity. In order to model such a context in our system, privacy values have to be defined first for different levels of perceived sensitiveness to information disclosure. In this case, Angela can define three different privacy levels: (i) *transparent*, for context information which can be shared without any privacy concern; (ii) *protected*, for potentially sensitive information which can be disclosed at her discretion, depending on the context; and (iii) *private*, for sensitive information which should never be disclosed. Those privacy levels are mapped to numeric values between 0 and 1. The next step is to model the context information as a context state, for instance as follows:

$$C_1 = \{location, phone\ number, address\}.$$

The context states are assigned the privacy values based on the rules defined in the system. A sample rule is given below.

```
1 rule: work at home;
2 if ( location == home ) and ( activity == working ) then set the privacy level of C1 to private;
```

The rule is intended to hide personal information (e.g., home phone number and address) to other people (e.g., customers) when the user is teleworking from home.

5.2.1. Hybrid approach

Let us assume that a request $C_r = \{e_1, e_2, e_3\}$ is sent to Angela by some other user, and that the associated privacy values are $w(e_1) = 0.4$, $w(e_2) = 0.3$, and $w(e_3) = 0.2$. Let us also assume that $\pi = 0.6$. Then we can evaluate Expression (1):

$$\sum_{j=1}^{3} w(e_j, t) = 0.4 + 0.3 + 0.2 = 0.9 \leq 0.6$$

which is not satisfied. Therefore, we proceed to greedily solve the optimization problem as defined by Expressions (2)–(3). Our approach is based on a decision tree and works as follows. First, the context elements are sorted in ascending order of their weights; in this case, sorting leads to (e_3, e_2, e_1). Then, the decision tree is built, where each node is a tuple of the form $<$*context element being evaluated*, *residual privacy value*, *set of selected context elements*$>$. Here, the first item is the context element, which is evaluated at the current step; the second one is the remaining privacy value, i.e., the privacy value which can still be satisfied given the current choice of context elements; finally, the last item is the set of elements which have already been selected by satisfying Expression (3). At each step, one context element is picked from the ordered list and evaluated. The context element is either selected as one of the context elements in C_s or dropped. The different options correspond to different paths in the tree. Note that an element can be added to the list of selected elements only if its weight is lower than the residual privacy value. The final set C_s is selected as the one with highest cardinality, and then the lowest residual privacy value.

The decision tree produced by the hybrid algorithm for the sample privacy values introduced above is illustrated by Fig. 2. In this case, the solution $C_s = \{e_1, e_3\}$ is selected, since it is the one with the largest number of (distinct) context values, as well as the lowest residual privacy value for that number of elements. As a result, the hybrid approach informs the user of the session C_r by displaying the shared context elements in C_s and, at the same time, asks for the permission to share e_2.

[2] In both cases, we refer to information which characterizes the current state of the user, i.e., the phone number and the address through which the user is reachable at the time the context elements are queried.

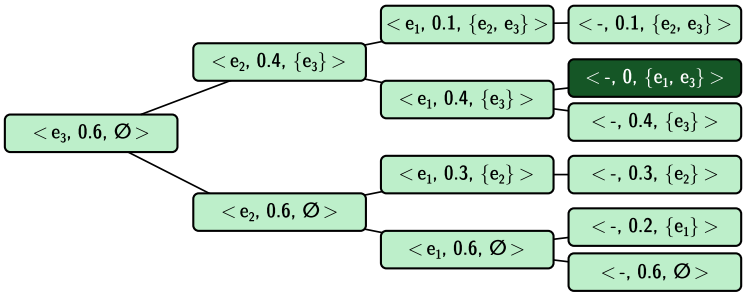

Fig. 2. Decision tree used by the hybrid approach.

Table 1
Privacy quantification based on user behavior.

Time (t)	$n(e_4(t), u_2)$	$\sum_j n(u_j)$	$\Pr(e_4(t) \mid u_2)$
t_0	0	4	0
t_1	0	5	0
t_2	1	6	0.167
t_3	2	7	0.286
t_4	3	8	0.375
t_5	4	9	0.444

5.2.2. Approach based on user behavior

Let us now consider the *loyal customer* use case, and assume that e_4 is the context element associated with the resources for internal use at the company. Now we focus on the interactions between Angela and the customers she is supporting, which include Bill (denoted as user u_2). These interactions, as well as the status of the system, are reported in Table 1.

Bill asks Angela for context information, which includes[3] the context element e_4. Let us assume that e_4 has already been requested 4 times by other users at a certain time t_0. Then, at time t_1, Bill requests a context which involves e_4, and Angela denies it, since she does not trust Bill sufficiently. The status of the system is updated accordingly: the number of times e_4 is requested is increased by one while the number of times Angela has shared e_4 with Bill is not changed and remains zero (Table 1). Some time later, Bill asks again Angela for context information involving e_4. Now Angela acknowledges that Bill is a loyal customer, and grants access to e_4 at time t_2. The same happens for the next requests until time t_4.

Now, let us concentrate on the request made by Bill at time t_5, and let us assume that it is in the form $C_5 = \{e_4, e_5\}$. According to the previous interactions, the sharing probability of e_4 is 0.444. The context element e_4 can be automatically shared or not, depending on the probability associated with $e_5(t_5)$, and according to Expression (3). Hence, e_4 is automatically shared if $\Pr(e_5(t_5) \mid u_2) \leq 0.444$.

5.3. Implementation considerations

In this section, we elaborate on the system implementation in a corporate scenario. In the following, we will consider the different subsystems introduced in Section 3.1 and detail how they could be realized with specific reference to the considered use cases.

In order to expose contextual information about the employer and their environment, some sensors would be needed in the first place. In a corporate scenario, it is actually possible to characterize the workplace through different modalities. For instance, rooms are usually equipped with motion sensors for lighting and intrusion detection purposes. More recently, buildings have been instrumented with metering devices for fine-grained characterization of energy consumption. Both categories allow to characterize the presence of an employee in a specific location (e.g., their office) with some accuracy. The location and the activity of the employees can be more accurately characterized by exploiting personal devices such as smart phones. This allows to track the movements of the employers or even the presence at different locations, e.g., branches of the same company or even the employer's home when tele-working.

The privacy manager would be implemented at the mobile device of the employer, namely, a laptop or a smartphone. Clearly, the actual location would depend on the device used to accomplish most of the working activity. The smartphone could also be used as the end-user interface with the system. In this case, the different components would need to be distributed across several devices. In a corporate environment, it is possible to have some functionalities implemented as services running on a remote host. This server could be hosted within the premises of the company or could be part of the corporate network, if the related functions need to be shared across multiple non co-located branches.

[3] For clarity, we will focus on the single context element e_4 in the following discussion.

The context manager would also be implemented at the mobile device. As for the actual rule evaluation engine, a simple Event-Condition-Action approach could be used [22]. Additional expressivity could be obtained by incorporating ontologies and context grammars at the cost of higher resource demands.

The proposed system could easily be integrated into the typical workflow of an employee. For instance, a personal information management software could expose a user interface to define different privacy levels and associate them to the context elements [28]. Such a software component could be realized, for instance, as an add-on for the Microsoft Outlook software, and would run as part of the application support subsystem. In general, any emailing or tele-conferencing software would be a suitable target for integration with the privacy preservation system.

6. Evaluation

In this section, we will evaluate the performance of the proposed approaches to privacy preservation in pervasive environments through simulation. In addition to the hybrid and the user behavior schemes, we will also consider the system-centric approach introduced in [11] for comparison purposes.

In order to evaluate the different schemes, we will consider the following metrics:

- *Sharing efficiency*, defined as the ratio between the number of context elements automatically shared (i.e., $|C_s|$) to the total number of context elements in a request (i.e., $|C_r|$). We consider the average value of the sharing efficiency for different requests during a simulation.
- *Number of queries*, defined as the average number of prompts to the user (i.e., $|C_r| - |C_s|$).
- *Number of operations*, defined as the average number of operations needed to evaluate whether a context element can be shared or not. For the system-centric approach, they are mapped to the number of operations needed to update the Context-Privacy Graph (CPG) [11].

Clearly, the first two metrics quantify the user experience. Specifically, sharing a higher number of context elements automatically results in fewer explicit interactions with the user. Similar considerations apply to the number of queries. In the rest of the section we will first present the simulation setup, and then the experimental results.

6.1. Simulation setup

We evaluated the different schemes by using a custom simulator written in Java. We created context elements and rules according to a smart environment scenario [28]. In detail, we used the JBoss Enterprise Business Rules Management System [29] to create and manage the context-aware rulesets. We varied the number of privacy levels from 3 to 6, in order to incorporate even complex privacy settings. Furthermore, we generated 100 users according to different J2ME Mobile Information Device Profiles (MIDP) [30]. The size of the context requests was uniformly distributed between 0 and 50 for each session, and the global privacy setting π was between 0.5 and 0.9. Unless otherwise specified, the total number of context elements in the system was set to 100. The chosen parameters can model rather general scenarios with enough users and contextual information. With specific reference to the use cases presented in Section 5.1, they would represent a smart workplace scenario in a medium-scale corporate environment consisting of one or more stories across one or a few co-located buildings.

We performed different sets of experiments. In most of them, we varied the number of context elements in the request (i.e., $|C_r|$). In all cases, we considered several values for the global privacy setting π.

6.2. Simulation results

We show in Fig. 3(a) the sharing efficiency for the hybrid approach and that based on the user behavior as a function of the request size. In order to make the comparison more fair, we focused only on the results for $\pi = 0.9$. From the figure we can see that the approach based on user behavior has a higher sharing efficiency than the hybrid scheme. This happens because it is not actually constrained by the global privacy setting π, but it rather depends on user interactions. As a result, the scheme based on user behavior can better follow the preference of users as it changes during time, thus allowing more elements to be shared on the average. Also note that the hybrid approach obtains a good sharing efficiency (more than 75% on the average), which tends to increase as the size of the requests increases as well.

We now focus on the number of queries as a function of the number of sessions (i.e., the number of context requests), as illustrated in Fig. 3(b). In this case, we also consider different values of the global privacy setting. Clearly, both schemes are affected by the global privacy setting, and the approach based on user behavior always results in a lower number of queries. This is partly related to the sharing efficiency, as already shown above. It is worthwhile emphasizing that, when the context size is small, the number of queries is very low for both approaches, thus resulting in an acceptable number of prompts to the user, even when the global privacy setting is moderate (i.e., $\pi = 0.5$). We report in Table 2 detailed results for 4 sample users $u_2, \ldots, u_5$, randomly selected among all those available in the system, requesting contexts to user u_1.

Finally, we show in Fig. 4 the number of operations required by the proposed schemes and the system-centric approach in [11] as a function of the context request size. Note that, for the hybrid scheme and for that based on user behavior, the

Table 2
Detailed results for specific users in the simulation.

User	π	Sessions	$\lvert C_r \rvert$	Number of queries	
				User behavior	Hybrid
	0.5	10	4	1	2
	0.6	20	8	2	3
u_2	0.7	30	16	4	5
	0.8	40	21	6	9
	0.9	50	33	8	15
	0.5	10	4	3	6
	0.6	20	7	3	9
u_3	0.7	30	12	4	13
	0.8	40	19	5	17
	0.9	50	23	7	21
	0.5	10	5	1	3
	0.6	20	11	1	4
u_4	0.7	30	18	4	9
	0.8	40	22	7	11
	0.9	50	37	11	23
	0.5	10	6	2	1
	0.6	20	14	3	6
u_5	0.7	30	23	4	11
	0.8	40	31	6	18
	0.9	50	39	8	27

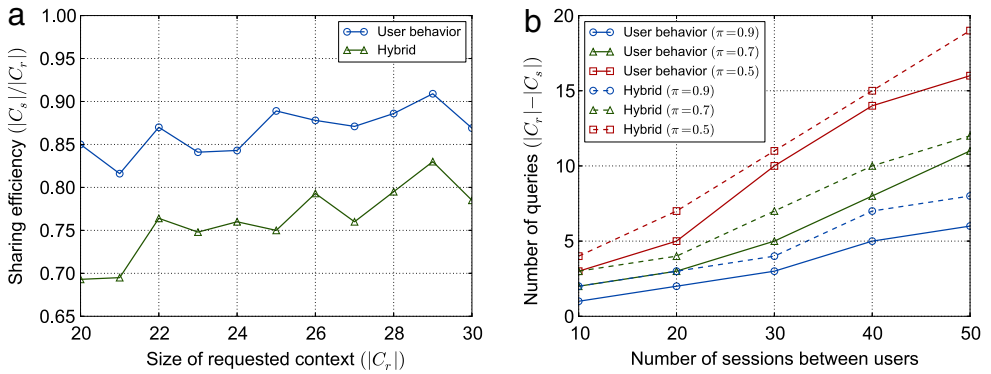

Fig. 3. Sharing efficiency (a) and number of queries (b) as a function of $|C_r|$.

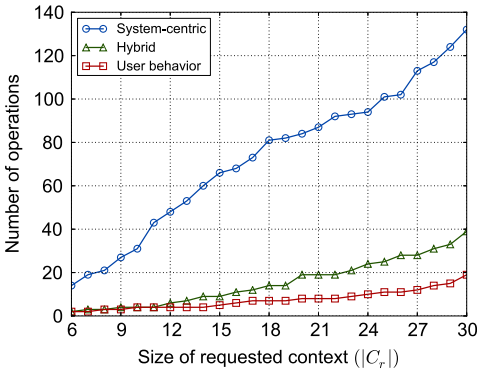

Fig. 4. Number of operations as a function of $|C_r|$.

number of operations is measured on the basis of the number of queries. The hybrid approach needs to build and parse the decision tree, while the one based on user behavior needs to go through the history of user interactions. Even though it is difficult to take all variables into consideration when comparing the different schemes, it clearly emerges that the overhead of system-centric approach is significant, especially when compared to the two schemes proposed in this article.

This happens because the Context-Privacy-Graph has to be re-evaluated (i.e., the different weights have to be reassigned) and its size increases quickly with the size of the requested contexts.

In conclusion, our schemes are quite efficient since they require a low number of prompts to the user, and they also have limited overhead due to the associated number of operations. The approach based on user behavior performs better than the hybrid scheme, since it can exploit the history of the user's sessions.

7. Conclusion

In this article we have proposed two context-aware schemes for privacy preservation in smart environments. In one case, we have exploited a hybrid approach to minimize user interventions in sharing context information. In the other case, we have leveraged user behavior to automate the sharing of context elements to requesting parties. We have considered a smart workspace application as a case study and we have presented two related use cases. We have also showed how our approaches can be conveniently mapped to the considered scenario. Finally, we have evaluated the performance of our solutions in terms of the user experience. Simulation results have shown that our schemes are effective in reducing the number of prompts and queries to the user, especially when interactions among different users are taken into consideration. Furthermore, our solutions are adaptive and have low resource demands; as a consequence, they are appropriate for dynamic environments and also suitable for implementation on personal devices as well as sensor nodes.

We are currently investigating some extensions as future work. Even though we have exploited a possible solution to incorporate user behavior into privacy preservation, our current approach is mainly based on the history of interactions between users. Since those interactions usually follow social patterns, our scheme can be extended to explicitly incorporate social behavior. Finally, we are planning to implement the proposed schemes on smart phones. We also intend to conduct additional experiments with the real implementation to better characterize the performance of the system.

Acknowledgments

This work has been partially supported by NSF Grants IIS-1064460 and CNS-1150192. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] J. Buckley, From RFID to the Internet Of Things: Pervasive Networked Systems, Tech. Rep., European Commission Directorate Network and Communication Technologies, 2006.
[2] U. Hansmann, L. Merk, M.S. Nicklous, T. Stober, Pervasive Computing: The Mobile World, second ed., Springer-Verlag, 2003.
[3] D.-H. Shin, Ubiquitous computing acceptance model & end user concern about security, privacy and risk, Int. J. Mobile Commun. 8 (2) (2010) 169–186.
[4] S. Poslad, Ubiquitous Computing: Smart Devices, Environments and Interactions, first ed., Wiley, 2009.
[5] D.J. Cook, S.K. Das, Smart Environments: Technology, Protocols and Applications, Wiley-Interscience, ISBN: 0471544485, 2004.
[6] D.J. Cook, S.K. Das, How smart are our environments? An updated look at the state of the art, Pervasive Mobile Comput. 3 (2) (2007) 53–73.
[7] M. Weiser, The computer for the twenty-first century, Sci. Am. (1991) 94–100.
[8] G.M. Youngblood, D.J. Cook, L.B. Holder, Managing adaptive versatile environments, Pervasive Mobile Comput. 1 (4) (2005) 373–403.
[9] B. Schilit, N. Adams, R. Want, Context-aware computing applications, in: IEEE Workshop on Mobile Computing Systems and Applications, 1994, pp. 85–90.
[10] A. Westin, Privacy and Freedom, fifth ed., Atheneum, 1967.
[11] G. Pallapa, N. Roy, S.K. Das, A scheme for quantizing privacy in context-aware ubiquitous computing, in: IET 4th International Conference on Intelligent Environments 2008, 2008, pp. 1–8.
[12] T. Heiber, P. Marrn, Exploring the relationship between context and privacy, in: P. Robinson, H. Vogt, W. Wagealla (Eds.), Privacy, Security and Trust within the Context of Pervasive Computing, vol. 780, Springer US, 2005, pp. 35–48.
[13] R. Beckwith, Designing for ubiquity: the perception of privacy, IEEE Pervasive Comput. 2 (2) (2003) 40–46.
[14] S. Lederer, I. Hong, K. Dey, A. Landay, Personal privacy through understanding and action: five pitfalls for designers, Pers. Ubiquitous Comput. 8 (6) (2004) 440–454.
[15] A.R. Beresford, F. Stajano, Location privacy in pervasive computing, IEEE Pervasive Comput. 2 (1) (2003) 46–55.
[16] J.I. Hong, J.D. Ng, S. Lederer, J.A. Landay, Privacy risk models for designing privacy-sensitive ubiquitous computing systems, in: DIS '04: Proceedings of the 5th Conference on Designing Interactive Systems, 2004, pp. 91–100.
[17] C. Dwork, Differential privacy: a survey of results, in: M. Agrawal, D. Du, Z. Duan, A. Li (Eds.), Theory and Applications of Models of Computation, in: Lecture Notes in Computer Science, vol. 4978, Springer, Berlin, Heidelberg, 2008, pp. 1–19.
[18] U. Hengartner, P. Steenkiste, Avoiding privacy violations caused by context-sensitive services, Pervasive Mobile Comput. 2 (4) (2006) 427–452. Special Issue on PerCom 2006.
[19] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, N. Triandopoulos, AnonySense: privacy-aware people-centric sensing, in: Proc. of the 2008 International Conference on Mobile Systems, Applications, and Services (MobiSys), 2008, pp. 211–224.
[20] E. De Cristofaro, A. Durussel, I. Aad, Reclaiming privacy for smartphone applications, in: 2011 IEEE International Conference on Pervasive Computing and Communications (PerCom), 2011, pp. 84–92.
[21] K.R. Raghavan, S. Chakraborty, M. Srivastava, H. Teague, OVERRIDE: a mobile privacy framework for context-driven perturbation and synthesis of sensor data streams, in: Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones, PhoneSense '12, 2012, pp. 2:1–2:5.
[22] N. Roy, G. Pallapa, S.K. Das, A middleware framework for ambiguous context mediation in smart healthcare application, in: WIMOB '07: Proc. of the Third IEEE Conference on Wireless and Mobile Computing, Networking and Communications, 2007, p. 72.
[23] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, first ed., Springer, 2004.
[24] S. Martello, P. Toth, Knapsack problems: algorithms and computer implementations, John Wiley & Sons, Inc., New York, NY, USA, ISBN: 0-471-92420-2, 1990.

[25] B. Johanson, A. Fox, T. Winograd, The interactive workspaces project: experiences with ubiquitous computing rooms, IEEE Pervasive Comput. 1 (2) (2002) 67–74.
[26] C. Le Gal, J. Martin, A. Lux, J.L. Crowley, Smart office: design of an intelligent environment, IEEE Intell. Syst. 16 (4) (2001) 60–66.
[27] N. Rump, K. Geramani, J. Baldzer, S. Thieme, A. Scherp, J. Krsche, J. Meyer, Potentials of pervasive computing in highly interactive workplaces, in: 10th ISPE International Conference on Concurrent Engineering: Research and Applications, 2003, pp. 733–739.
[28] G. Pallapa, A privacy enhanced situation-aware middleware framework for ubiquitous computing environments, Ph.D. Thesis, The University of Texas at Arlington, http://hdl.handle.net/10106/4867, 2010.
[29] The Drools Project, JBoss Enterprise Business Rules Management System, http://jboss.org/drools, 2010.
[30] Sun Microsystems/Oracle, Mobile Information Device Profile, http://java.sun.com/products/midp/overview.html, 2010.