Dynamic Privacy Management: a Plug-in Service for the Middleware in Pervasive Computing

Dan Hong
Department of Computer Science
Hong Kong University of Science and Technology
Hong Kong, China
csdhong@cs.ust.hk

Mingxuan Yuan
Department of Computer Science and Technology
Xi'an Jiaotong University
Xi'an, Shaanxi, 710049, China
harry_yuan@ei.xjtu.edu.cn

Vincent Y. Shen
Department of Computer Science
Hong Kong University of Science and Technology
Hong Kong, China
shen@cs.ust.hk

ABSTRACT

Context-aware applications can better meet users' needs when sensing agents installed in the environment automatically provide input relevant to the application. However, this non-intrusive context usage may cause privacy concerns since sensitive user data could be leaked to unauthorized parties. Therefore, data privacy protection becomes one of the major issues for context-aware applications. In this paper, in order to provide services based on various levels of privacy concerns, we extend the Platform for Privacy Preferences of W3C and define a specification for representing user privacy preferences for context-aware applications. We also propose a privacy infrastructure, which could be installed as a plug-in service for middleware supporting context-aware applications. This infrastructure enables the middleware to automatically generate a privacy policy and the user preference file according to the current context. The middleware simply matches these two files to decide whether to proceed with the application. We demonstrate the efficacy of this approach through a prototype implementation.

Categories and Subject Descriptors

K.4.1 [**Computers and society**]: Public Policy Issues—*Privacy*; H.1.2 [**Models and principles**]: User/Machine Systems—*Human factors*

General Terms

Design, Human Factors

Keywords

privacy, context, P3P, pervasive computing, middleware

1. INTRODUCTION

Sensing technologies have made significant progress in recent years. Their availability in a pervasive computing environment can provide the context needed for certain applications. Context-awareness enhances these applications by providing more environmental parameters, such as location, time, activities, and so on, as automatic inputs to the application. This automation reduces manual interaction between users and computing devices, and therefore context-aware applications are gradually gaining acceptance by most users [4] [6] [9] [18].

However, users are concerned about possible leakage of privacy information. This problem has been pointed out when the "active badge" system [17], the first context-aware application, was tested at the Xerox Palo Alto Research Center. Further studies [3] [11] confirm that privacy is a difficult design issue, which could become a major user concern. A privacy data owner wishes to know not only who can access the privacy data, but also what types of data are collected and how are they used.

The pervasive computing environment makes privacy data management different from other application domains. The ubiquitous sensors are unobtrusively deployed and work all the time. Their invisibility, which leads to a very exciting user experience, introduces the important privacy issue at the same time. This non-intrusive nature of pervasive computing causes user concern because there is less information indicating whether privacy data are collected and used. It is not appropriate to send a permission request to the privacy data owner each time such data are used. Another problem is that user preferences may change dynamically when the context changes. For example, the purpose of a student who goes to a coffee shop may affect his privacy decision. If he goes there to meet with his friends, he is willing to let his friends know his location. However, if he goes there to study, he might not wish to be disturbed and would prefer that nobody knows his location. In this example, the location data protection relies not only on the current location but also the purpose. This dynamic privacy data protection issue has not been addressed by previous research.

The Privacy Policy Preference Platform (P3P) recommended by the W3C [15], is the first "social" protocol defined in an XML-based language to address the privacy issue technically [2]. P3P enables websites to translate their privacy practices into a standard, machine-readable format that can be retrieved automatically and interpreted by a user's browser [1]. P3P supports privacy management effectively when users browse the Web where the context is basically fixed. The specification is very useful for concise presentation of the privacy policy in terms of purpose, recipient, retention, remedy, and so on. In this paper we define an extension of P3P to support context-aware applications. We use the extension space in P3P to specify policies relevant to context-aware applications. To represent user preferences we also define another markup language which helps machine processing within the middleware.

We propose a privacy management infrastructure which could easily be plugged into the middleware, which is the intermediary between applications and sensing agents. The middleware manages contexts pushed from sensing agents, sends appropriate context to an application, or triggers sensing actions when an application needs more context data which is not available in the context database. One advantage of using middleware is that it can provide centralized context management for all applications. These applications get context from the middleware indirectly instead of communicating with thousands of sensing agents individually. This would speed up the development of context-aware applications. Our infrastructure enables the middleware to make decisions dynamically based on real-time computed results, which are dependent on both the users' preferences and the context in the pervasive environment. Moreover, our work not only offers flexibility to users for modifying privacy policy but also enables users to extend their own privacy interests. This means that users can define their own pervasive environment parameters which are part of later computing decisions. As a result, the middleware can provide more personalized services.

The rest of the paper is organized as follows. Section 2 describes an overview of privacy management architecture. In Section 3, we define our privacy markup language. Section 4 presents how our automatic privacy policy infrastructure works. We discuss the benefits of our current results in Section 5 and compare them with related work in Section 6. Finally, we present our conclusions in Section 7.

2. MIDDLEWARE OVERVIEW

Our privacy management is based on the middleware architecture, which is shown in Figure 1. The middleware works between the sensing agents and applications. At the bottom layer, sensing agents continuously monitors the environment and send context data to the middleware. Context Collector receives all data and stores them in the Context Database for later retrieval. At the top layer, an application uses Communication Support to communicate with the middleware to get context data,. When the middleware receives a request from Context Dispatcher, Application Manager searches the Context Database and sends back the data to the application with the scheduling help of Context Dispatcher.

If an application wants to collect some privacy data, then it uses Application Privacy Policy Generator to automatically generate a new privacy policy file according to the current context. It then sends the policy and the context query (the same query as the one when there is no privacy service) together to the middleware through Communication Support to the Application Manager. Application Manager forwards the privacy policy to Privacy Data Mediator to justify the access rights of the application and privacy data inquirer. Based on the privacy data owner's user preference file and the context from the pervasive environment, Privacy Data Mediator generates a context-aware user preference and then checks if the application has the right to access the context. After negotiating with Privacy Data Mediator, Application Manager either searches the context from the Context Database or rejects the request directly.

Traditional Role Based Access control (RBAC), which gives the same access rights to people in the same group, allows privacy data owners to set up their own preference. However, it is not enough for the pervasive environment since there are more concerns than just people. In Privacy Data Mediator, we enhance RBAC to multiple dimensions which could be defined by privacy data owners.

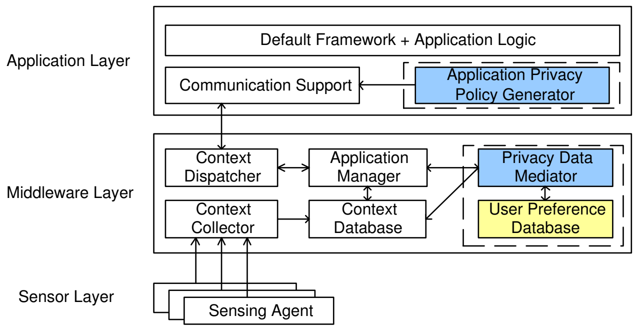


Figure 1: Overview of middleware architecture

From Figure 1, we can find out that Privacy Data Mediator works independently in the middleware. If a middleware does not already support privacy management, it is easy to plug in a Privacy Data Mediator without changing the existing implementation.

3. PRIVACY MARKUP LANGUAGE

3.1 Privacy policy

P3P is a labeling protocol which allows services and individual users to come to agreement on the release of personal data [2]. Figure 2 illustrates an example of a P3P policy. Alice collects current location information, which is indefinitely kept and might be forwarded to her partners in order to finish the current service. The access is given to contact information as well as to certain other identified data.

Since P3P is designed for websites to collect user personal data through the browser, it needs to be extended for context-aware and pervasive environments [1]. In our approach, we use the same syntax in P3P and make an extension for the pervasive environment. When an application collects some privacy data from the middleware, it provides as much information as it can in order to help privacy data owners make the decision automatically. All the related information could be included in a privacy policy file. A policy file defines:

- **What**: the privacy data that the application collects. This is the first factor privacy data owner concerns when there is a query. The queried context helps to reduce manual input in meeting users' needs during an interaction.

```
<?xml version="1.0" encoding="UTF-8" ?>
<POLICIES xmlns="http://www.w3.org/2002/01/P3Pv1">
    <POLICY name="global" >
        <ENTITY>
            <DATA-GROUP>
                <DATA ref="#business.name">Alice</DATA>
            </DATA-GROUP>
        </ENTITY>
        <ACCESS><contact-and-other/></ACCESS>
        <STATEMENT>
            <PURPOSE><current/></PURPOSE>
            <RECIPIENT><ours/></RECIPIENT>
            <RETENTION><indefinitely/></RETENTION>
            <DATA-GROUP>
                <DATA ref="#dynamic.miscdata">
                    <CATEGORIES><location/></CATEGORIES>
                </DATA>
            </DATA-GROUP>
        </STATEMENT>
    </POLICY>
</POLICIES>
```

Figure 2: A P3P policy example

- **Why**: the purpose for which the application collects the data. This is the most important item that affects the privacy data owner's decision.
- **Who**: the people who receive the personal information, could also be referred to as "privacy data inquirer". It could be either the user of a context-aware application or the application itself. Privacy data owners are willing to provide data only when they trust.
- **When**: the time when the privacy data inquirer requests context from the middleware.
- **Whom**: the privacy data owner. Different users may have different privacy preferences. According to this information the middleware knows whose user preference it should check in User Preference Database.
- **Where**: the location the application takes place.

For a context-aware application, some items ("What" and "Why") discussed above are quite stable. For a single application, no matter who uses this application, or when and where a user uses it, these items always stay the same. These first two points have already been well-described by P3P elements <PURPOSE> and <DATA-GROUP> in <STATEMENT> respectively.

Not all the items are well-defined in P3P. P3P uses <ENTITY> to describe the business communities who collect the data. Obviously, it is not enough to represent the "Who" item, which includes both the privacy data inquirer and the application. Therefore, we extend the element <ENTITY> as follows:

```
extension = "<EXTENSION>"
            "<DATA-GROUP>"
            "<DATA ref=`#application.name`/>" PCDATA "</DATA>"
            "</DATA-GROUP>"
            "</EXTENSION>"
```

The last three items, "When", "Whom", and "Where", are not specified in P3P. We find out that the declaration about context data collection in P3P is too simple for privacy data owner to make decisions. For example, one user Alice uses a Find-a-friend application to look for all the friends on her friends list. However, her friends may have a privacy concern related to Alice's location. Therefore, the policy should contain the identifiers about whose locations are collected, when the context is collected, and where Alice is. However, it is hard for privacy data inquirers to denote the information in original P3P privacy policies. In order to adapt to this new feature of privacy data collection, we add one more <EXTENSION> as the child of the <POLICY> element.

```
extension  = "<EXTENSION>"
             "<DATA-GROUP>"
             "<DATA ref=`#extra.time`/>" PCDATA "</DATA>"
             "<DATA ref=`#extra.whom`/>" PCDATA "</DATA>"
             "<DATA ref=`#extra.location`/>" PCDATA "</DATA>"
             "</DATA-GROUP>"
             "</EXTENSION>"
```

In order to share context all over the world, we use "Coordinated Universal Time" to indicate time. With the help of the new extension of <ENTITY> and <POLICY>, the privacy policy file could provide a rich context for the middleware to automatically make the decision for privacy data owners.

```
<?xml version="1.0" encoding="UTF-8" ?>
<POLICIES >
    <POLICY name="global" >
        <ENTITY>
            <EXTENSION>
                <DATA-GROUP>
                    <DATA ref="#application.name">Find-friend</DATA>
                </DATA-GROUP>
            </EXTENSION>
            <DATA-GROUP>
                <DATA ref="#business.name">Alice</DATA>
            </DATA-GROUP>
        </ENTITY>
        ......
        </STATEMENT>
        <EXTENSION>
          <DATA-GROUP>"
              <DATA ref="#extra.time"/>2005-03-12T19:20+01:00</DATA>
              <DATA ref="#extra.whom"/>Bob</DATA>
              <DATA ref="#extra.location"/>Room4201 </DATA>
          </DATA-GROUP>
        </EXTENSION>
    </POLICY>
</POLICIES>
```

Figure 3: A policy example after extending P3P

For our example, Alice uses the Find-a-friend application, and queries the location context of her best friend Bob. Figure 3 is the example of a privacy policy file after we extend P3P. Note in the <ENTITY>, we describe the application name (Find-a-friend) as an <EXTENSION> element. Alice's current location (Room4201) and query time (when she wants to know about Bob at 2005-03-12T19:20+01:00) are defined in the <EXTENSION> element of <POLICY>.

3.2 User preference file

In order to coordinate and match with privacy policy easily, the user preference file uses similar syntax with privacy policies. In a context-aware application there may be too much private context. According to P3P v1.0 [15], there are already about eighty privacy data items and the number keeps growing. So it is not applicable for users to define each privacy protection mechanism one by one. In our approach, we provide the privacy protection based on the <CATEGORIES> concept. There are seventeen categories defined in P3P, such as <PHYSICAL/>, <PREFERENCE/>, <LOCATION/> and so on. We define the category-based preference representation below:

```
preference = "<PREFERENCES >"
             1*categories block
             "<PREFERENCES >"

categories  = "<CATEGORIES >"
              1*category block
              "</CATEGORIES >"

   category = "<"category name ">"
              1*application groups block
              1*user groups block
              * extension
              "</"category name ">"
```

Each preference file has several categories. The category names are the same as the definition in P3P. For each category we basically have two blocks (could be extend by extension): "application groups" block and "user groups" block, both of which are used to define the group-based access rights. This is because without a completely accurate grouping mechanism (or some manner of collapsing categories in a meaningful way), few users would be able to correctly categorize a situation without errors [2]. Each block contains several group definitions according to the same category. So far, the privacy protection is still RBAC, which is too simple for pervasive computing. So we extend these two blocks by adding the "time" block. By doing so, we allow the privacy data owners to manage their preferences in more detail by time, which is the common factor that affects privacy decisions. There are two kinds of time blocks. One is the "permission" time, which denotes a restricted time period when data in this category could be accessed. The other one is the "query" time, which denotes the time period when the query is allowed. In different time periods, privacy data owner could have totally different preferences. We use <INCLUDE> and <EXCLUDE> to denote the time period when access is allowed or denied, respectively. The exact access rights (purpose, dispute, and retention) are defined in the access statement as it is defined in [15]. In different time periods, a privacy data owner could have totally different preferences by specifing different time blocks.

```
          time = "<TIME  ref=time.string>"
                 1* include block
                 1* exclude block
                 access statement
                 "</TIME>"

       include = "<INCLUDE >"
                 time period; defined in Time definition lists
                 "</INCLUDE >"

       exclude = "<EXCLUDE >"
                 time period; defined in Time definition lists
                 "</EXCLUDE >"

access statement = access
                   [disputes-group]
                   purpose
                   rentention
```

Figure 4 is a simple example of a user preference file. It allows people in the FRIENDS list to access privacy data in workingtime (except classtime) if the purpose of collecting this data is for the completion of current activity and the inquirer's location is Room4201. This preference file also allows applications in the MY FAVORITE list to access the location data during holidaytime if the application uses these data for contact purpose and promises that it does not retain them. The attributes workingtime, classtime, holidaytime, MY FAVORITE and FRIENDS are defined in this user's time definition list and users list, respectively (discussed in 4.2.1).

```
<?xml version="1.0" encoding="UTF-8" ?>
<PREFERENCES >
    <CATEGORIES>
        <LOCATION>
            <USER-GROUPS>
                <FRIENDS>
                    <TIME ref="#time.permission">
                        <INCLUDE><workingtime/></INCLUDE>
                        <EXCLUDE><classtime/></EXCLUDE>
                        <ACCESS><contact-and-other/></ACCESS>
                        <PURPOSE><current/></PURPOSE>
                        <RETENTION><indefinitely/> </RETENTION>
                        <RECIPIENT><ours/> </RECIPIENT>
                        <EXTENSION>
                            <DATA-GROUP>
                                <DATA ref="inquirer.location">ROOM4201</DATA>
                            </DATA-GROUP>
                        </EXTENSION>
                    </TIME>
                </FRIENDS>
            </USER-GROUPS>
            <APPLICATION-GROUPS>
                <MY FAVORITE>
                    <TIME ref="#time.permission">
                        <INCLUDE><holidaytime/></INCLUDE>
                        <ACCESS><contact-and-other/></ACCESS>
                        <PURPOSE><contact/></PURPOSE>
                        <RETENTION><no-retention/></RETENTION>
                        <RECIPIENT><ours/> </RECIPIENT>
                    </TIME>
                </MY FAVORITE>
            </APPLICATION-GROUPS>
        </LOCATION>
    </CATEGORIES>
</PREFERENCES>
```

Figure 4: User preference file example

4. PRIVACY COMPONENTS

From Figure 1, we can see that if the middleware wants to support privacy management, we need to modify both the application layer and the middleware layer. In the application layer, the privacy data inquirer provides its privacy data collection policy via the Application Privacy Policy Generator. The middleware layer provides privacy management service to the privacy data owner through the Privacy Data Mediator by sending back the context or rejecting the request.

4.1 Application layer privacy component

In the application layer, the only component is Application Privacy Policy Generator. From the discussion in 3.1, we can see that the privacy policy is different in different situations. The main function is to automatically generate a new extension P3P privacy policy file for the application according to the current context (e.g., privacy data inquirer and current location) and some input from the privacy data inquirer (e.g., privacy data owner and query time).

The context, which an application is interested in, has similar properties. For example, the Find-a-friend application cares only about the privacy data owner's location data. Therefore, the application could have some basic privacy policy which does not need a dynamic environment context. Then when a privacy data inquirer uses the application, what Application Privacy Policy Generator needs to do is to extend the basic privacy policy to the format we defined in 3.1.

4.2 Middleware layer privacy components

In a context-aware application the same user might have different preferences depending on the context. The traditional RBAC does not support this requirement since the privacy data owner could group several people together and give this group the same right to access privacy data. Such groupings cannot address multiple contexts, such as the time and purpose of access.

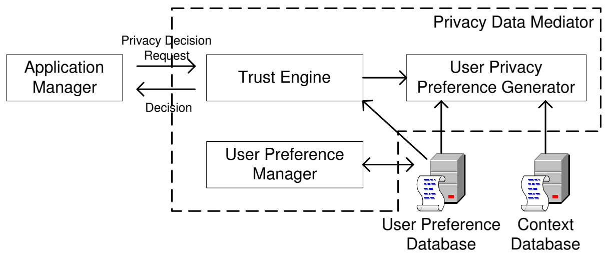


Figure 5: Overview of privacy management service

In our approach, we take care of each component that is related to decision making about the privacy data. Privacy Data Mediator, which is illustrated in Figure 5, basically has three components: Trust Engine, User Privacy Preference Generator, and User Preference Manager. Trust Engine is a service provided to Application Manager when an application requests some contexts related to a certain subject, whose information might be accessed. When a privacy decision request is received, Trust Engine calls the User Preference Generator, which automatically generates a user preference file according to the current context. User Preference Generator first checks the User Preference Database, finds the specific preference file for the privacy data owner, searches for some related context in Context Database, and automatically generates a real-time preference file based on the preference file and the context. Comparing the privacy policy and user preference, Trust Engine makes the final decision whether to accept or reject the request. User Preference Manager is another privacy service, through which users could identify their own interested contexts, and set up privacy preferences for these contexts.

4.2.1 User Preference Database

User Preference Database stores files that are related to user preferences. It contains five types of files.

- User preference files: define category-based user preferences about privacy data. These are the most important files, representing privacy data owners' view regarding context.
- Time definition lists: define alias names of different time periods for each user, such as workingtime, which may be defined as 8.00am–5.00pm from Monday to Friday.
- User group lists: define group information for RBAC. Each group of people has the same access right to one category. Each group has one or more people and each person could belong to several groups.
- Application group lists: group applications, which are always given the same access rights, in one cluster
- Privacy protection level: define privacy protection quality. Privacy data owner can use User Preference Manager to modify this.

Time definition lists, user group lists, and application group lists are references for user preference files. In the example of user preference file in Figure 4, we can see that by breaking these files away, the user preference files are much easier to read not only by machines but also by people. How to implement the reference files depends on the developers, and this does not affect the interoperability between the applications and the middleware.

4.2.2 User Preference Generator

One feature of pervasive computing is that everything depends on dynamic context variations. As a result, privacy preferences might be different due to context changes in context-aware applications. Since the user preference file contains multiple preferences, which are suitable for diverse situations, it is necessary for the middleware to find the preference for the current situation. This is the reason that User Preference Generator is needed. Its main function is to find out which preferences are appropriate for the privacy data owner in the current context and to generate a new preference file based on existing preferences.

The generator basically achieves this goal in four steps. The first step is to locate the user's preference file by "Whom" value from the privacy policy. For convenient purpose, if the privacy data owner does not modify his own privacy preferences, the generator uses the default privacy policy to protect his privacy data. Second, the generator checks the privacy preference declaration based on categories, which are provided in the privacy policy. Then, based on the related context information defined in the user preference file, the generator requests the context information from the context database. Finally, the generator removes the unsuitable preference items, which are in conflict with the related context parameter, and creates a new user preference file as defined in 3.2 , which is suitable for the current situation.

From the discussion above, we can see that not all the information that affects the privacy decision is provided by the application and the generator meanwhile might query the context database directly in order to get the related context. The reason not everything is provided by an application is that the generator has more rights to access privacy data, which is not accessible by the application. This solves the problem that an application requests context when the privacy decision about this context is related to some contexts which the application has no right to access.

4.2.3 Trust Engine

After the User Preference Generator finishes its work, a new preference file is passed to Trust Engine. Though this file has removed any unrelated items, it still has redundant information. This is because users and user groups (or applications and application groups) are not always in one-to-one mapping. A privacy data inquirer might be in several groups which have different access rights to the same category. For example, Alice is in the close friends group as well as in COMP101 classmates group. Bob gives the location data access right to the close friends group at any time but gives the COMP101 classmates group another access right only during the class time of COMP101. So if Alice queries the location information via an application (assuming that the application is not related to location data access), after the User Preference Generator process the file, Trust Engine receives an ambiguous file from the User Preference Generator.

Trust Engine accepts or rejects the context request by matching the preference file with the policy file. Trust Engine provides different levels to protect privacy data.

- **Maximum protection:** This gives the strictest access to the privacy data. If one preference does not match the policy, the request is rejected.
- **Minimum protection:** This is on the contrary to Maximum protection and gives broadest access to the privacy data. If one preference matches the policy, then it accepts the request.
- **Privacy-level:** Privacy level [10] is a function relating to activity, time, location, and so on. The user's subjective values determine the actual level of preferred privacy. If a privacy level value computed from the real-time user preference file is greater than this level then the request is accepted and vice versa. The privacy data owner could define the protection level in the privacy level file and modify it through User Preference Manager.

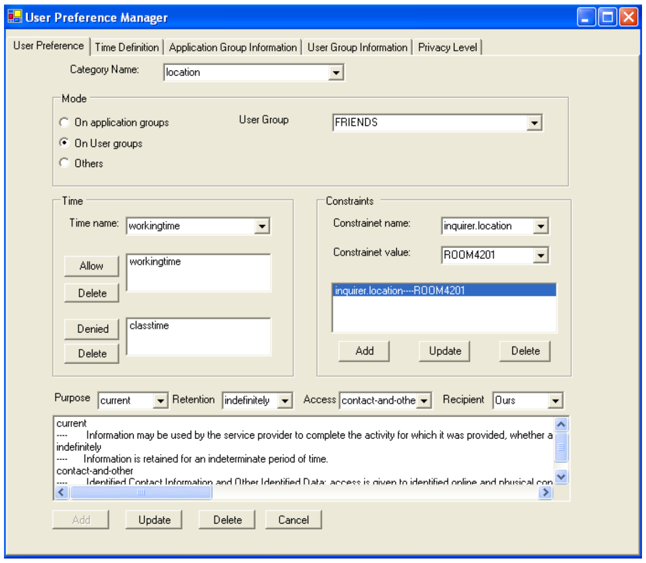
Trust Engine provides services for Application Manager when some personal contexts are required. Instead of making acceptance and total rejection decision, there are some other choices, especially when there is a conflict among multiple preferences. One possible solution is to record all the conflict and ask the privacy data owner to make the decision directly. This method is a little bit inconvenient for the privacy data owners since it might frequently interrupt privacy data owner's normal work. It could be worse if the privacy data owner is not connected to the middleware and there is no way for the privacy data owner to choose to accept or reject the request. Another possible solution is for the middleware to send some fuzzy message, such as "The context requested is not in the database", instead of a full rejection. This is because privacy data owners may not like to let others know that their request is rejected although they really wish to do so. Sending such fuzzy messages needs to be considered during middleware development.

4.2.4 *User Preference Manager*

Privacy data owners can add, update, and delete their privacy preferences by User Preference Manager. One criticism of P3P is that its vocabulary and structure may be too complex for naive users to use [7]. From the discussion in 4.2.1, User Preference Manager should provide a user-friendly interface for privacy data owners to manipulate five kinds of files. Figure 6 is a possible layout of the User Preference Manager.

The User Preference Manager console has a multiple-tab interface and each tab corresponds to one data type. In the Time Definition, Application Group Information, and User Group Information tabs privacy data owners define their own alias name for each item. Also in the User Preference tab they could specify their preferences based on the category concept. Most privacy data owners do not have knowledge about the privacy policy and therefore it is hard for them to understand the options in <PURPOSE>, <RETENTION>, <ACCESS>, and <RECIPIENT>. To assist the user, we provide readable explanations about preference manipulations at the bottom of the panel.

From Figure 6, we can see this interface is easy to use on a desktop computer. However, privacy data owners sometimes may need to use PDAs or cell phones to revise their preferences. The layout of the console should be designed to work with other input devices as well.



Category Name: location

Mode: ( ) On application groups [✓ On User groups] ( ) Others

User Group: FRIENDS

Time

Time name: workingtime

Allow: workingtime

Denied: classtime

Constraints

Constrainet name: inquirer.location

Constrainet value: ROOM4201

inquirer.location----ROOM4201

Purpose: current Retention: indefinitely Access: contact-and-othe Recipient: Ours

current
----	Information may be used by the service provider to complete the activity for which it was provided, whether a
indefinitely
----	Information is retained for an indeterminate period of time.
contact-and-other
----	Identified Contact Information and Other Identified Data: access is given to identified online and physical con

Figure 6: User Preference Manager console

4.2.5 *Implementation*

Xu *et. al.* proposed and implemented the middleware called Cabot [19] which could assist context transfer between applications and sensing agents. We use Cabot as the platform for our experiment where the sensing agents are simulated by software. We implement our privacy components as a Java plug-in service for Cabot except the user preference console, which is implemented in C#. We design a simple application called "Find-a-friend". In this application example, an inquirer queries the location information about his friends, and the friends manage their preferences through our console. While many of the friends reveal their locations, some of them may not wish to let the inquirer know their locations at that particular time and the query is rejected. From this experiment, we find privacy protection based on context using the plug-in works effectively.

This experiment on Cabot is just an example. Other middleware may have different implementations and architectures. However, implementing the Privacy Data Mediator as a plug-in should work well since we only need to have the right to access the Context Database of the middleware.

5. DISCUSSION

5.1 Extensibility

The reason users like to use a context-aware application is that the service is more user-centric. An individual user may have some special consideration that is different from others. It is very difficult to define a preference file that is suitable for everyone, everywhere, and at any time. So it is better that the Privacy Data Mediator can help privacy data owners extend their own interest. We care about the human factors during the interaction, and leave space for extension in the privacy policy and user preference file. The following is the extension element in TIME element.

```
extension ="<EXTENSION>"
            "<DATA-GROUP>"
            *("<DATA ref=`context-schema`/>" PCDATA "</DATA>")
            "</DATA-GROUP>"
            "</EXTENSION>"
```

Through User Preference Manager, privacy data owners could extend their own interest context parameters, which is related to privacy decisions, but does not disturb other design considerations. This enables a personalized privacy protection service for privacy data owners and provides a much more user-oriented service during the query process.

5.2 Flexibility

The markup language, which we discussed in Section 3, is designed to be a machine-processable interface for information exchange among the components. Therefore the components for privacy are relatively independent, which provides a lot of flexibility to the developers. They can use different languages (such as Java and C++) across different platforms. Moreover, in order to improve privacy protection, developers need only update one component and can leave other components as before. For example, if developers want to provide a higher privacy protection level to privacy data owners, they do not need to modify the User Preference Generator and only change the Trust Engine part.

From Figure 1, the privacy components are applicable to be inserted or become a replacement of the privacy box (if there is already a privacy management system) in the middleware. After achieving the goal, the middleware enables a privacy management service for all the applications that may query context from it. Moreover, when a new application or more sensing agents are added, the Privacy Data Mediator does not need too much modification except registering the application in the application list.

5.3 Security

Security is the basis of privacy. If there is no security protection in the middleware, there is no privacy because the communication channel could be attacked and all the data transmitted on this channel could be copied and used by third parties without permission. The security channel could be achieved either by encryption or by digital signature. The W3C XML encryption syntax and processing recommendation [16] specifies a process for encrypting arbitrary data (including an XML document) and representing the result in XML format. Moreover, the XML Key Management Specification (XKMS) [14], which is a joint work of W3C and IETF, could provide a security channel by digital signature. As a result, middleware developers could use such ways to enable secure data transmission.

In pervasive computing, not only does the communication channel affect security, where to store user preference files is also crucial. Users must make sure that the private context is stored in a safe place, as well as the user preference files. The middleware should not leak the private context to other illegal applications or users without the permission and allow other parties to modify the private context and preference files. In our infrastructure, this security is built on the trust relationship between the privacy data owner and the middleware. This relationship could be done by signing some contract or inheriting from some trustable parties. Nowadays, a context-aware application is normally designed for mobile devices, and in order to enlarge their market share mobile phone service providers try their best to work with other companies to offer extra exciting functions which are beyond basic mobile phone service. In this case, service providers act as Privacy Data Mediator in our infrastructure, while phone users could be privacy data owners (or privacy data inquirers in some applications). For example, one shopping center (privacy data inquirer) has a contract with a service provider to push discount information to phone users based on their private location information. It is clear that our infrastructure is applicable for this example and it is flexible for other companies to join in when they have contracts with the service provider.

5.4 Convenience

When people develop new technologies, they hope their work is compatible with existing applications. As one of the most popular applications on mobile devices (e.g. PDAs, mobile phones), mobile Web browsing should be supported by our infrastructure. By using P3P syntax, our privacy mediator fully supports the privacy policy in the existing Web and provides more powerful context-aware privacy protection when users browse the Web via their mobile devices.

Privacy feature has been a plugged-in service in browsers for a long time. However, it does not attract many users. The reason is that P3P-compliant user interfaces face the challenge of presenting often complex information about a privacy policy in a manner that is easily understandable by the user [1]. In this paper, we have proposed a possible user preference management interface. By picking out the complex items from the preference file and defining them in a separate file, users have more intuitive feeling about their preferences and have the ability to control their privacy data.

6. RELATED WORK

Many researchers have tried to address privacy issues in pervasive environments. Some research has focused on application-oriented privacy management. Fithian *et. al.* reported the privacy management interface design in Mobile Location-Aware Handleld Event Planner [6]. In [13] Roussos and Moussouri considered consumers perceptions about the grocery application and discussed about privacy and trust issues that may prevent its adoption. In these applications, privacy data owners can then manage their data since the data collected by a single application should be simple. However, this solution has two obvious problems: one is that users need to specify their own privacy preferences for each application, which is not convenient. Another problem is that there may be too much redundancy since each context need to be copied in each application in the pervasive environment.

Another approach is to offer centralized services. Based on privacy needs of users and application developers, Confab, a toolkit for the development of privacy-sensitive context-aware applications, is a framework which provides customized privacy mechanisms [8]. Lei *et. al.* [12] used a privacy engine, which provided Role Based Access Control (RBAC). RBAC reduces the cost of administration and is easily expressed by business practices. Convington *et. al.* extended the traditional RBAC to an "environment role", which could provide some privacy decisions based on pervasive environments [5]. The centralized privacy management service, which could easily be plugged into the middleware, enables privacy protection during interactions between users and all applications.

However, we notice that from previous work, "static" privacy management, as predefined by developers, is not suitable for dynamic pervasive environment. There could be some context in the pervasive environment, which may affect the final privacy policy decision of a specific user, is not taken into consideration by developers.

7. CONCLUSIONS

In this paper, we propose a markup language that is suitable for representation of both privacy policies and user preferences. The language enables a machine-to-machine privacy data collection process between the middleware and applications. Our Privacy Data Mediator, implemented as a plug-in for the middleware, plays a key role in enabling privacy data protection in a dynamic, extensible, flexible and convenient manner. As context-aware applications become more popular and the middleware becomes more widely deployed, the privacy components proposed in this paper will significantly reduce the number of human interactions needed for privacy data protection.

Mobile devices(HP IPAQ rx3715 PDAs) and location sensing equipment (Bluetooth GPS receivers) were purchased. A user-centric evaluation is being conducted in our campus through Cabot which has the dynamic privacy management component. Our next step is to evaluate when the right time is to interrupt users when there is a conflict between the policy and preference. We would also like to further investigate using Semantic Web to help users define their groups in order to simplify the interaction between users and the middleware.

Acknowledgements

This research was supported in part by Sino Software Research Institute grant SSRI01/02.EG14, "W3C Office" and by Research Grant Council grant AOE/E-01/99, "Information Technology for a 21st Century Hong Kong".

8. REFERENCES

[1] M. Ackerman, T. Darrell, and D. J. Weitzner. Privacy in context. *Human-Computer Interaction*, 16:167–176, 2001.

[2] M. S. Ackerman. Privacy in pervasive environments: next generation labeling protocols. *Personal and Ubiquitous Computing*, 8(6):430–439, November 2004.

[3] L. Barkhuus and A. Dey. Location-based services for mobile telephony: a study of users privacy concerns. In *9TH International Conference on Human-Computer Interaction (INCTERACT03)*, pages 709 –712, Zürich, Switzerland, July 2003. IFIP.

[4] T. Bohnenberger, A. Jameson, A. Krüger, and A. Butz. Location-aware shopping assistance: Evaluation of a decision-theoretic approach. In *Mobile HCI 2002*, pages 155–169, Pisa, Italy, 2002. Springer-Verlag.

[5] M. J. Covington, W. Long, S. Srinivasan, A. K. Dey, M. Ahamad, and G. D. Abowd. Securing context-aware applications using environment roles. In *Symposium on Access Control Models and Technologies*, pages 10–20, Chantilly, Virginia, USA, May 2001. ACM.

[6] R. Fithian, G. Iachello, J. Moghazy, Z. Pousman, and J. Stasko. The design and evaluation of a mobile location-aware handheld event planner. In *MobileHCI 2003*, pages 145–160, Udine, Italy, September 2003. Springer-Verlag.

[7] H. Hochheiser. The platform for privacy preference as a social protocol: An examination within the u.s. policy context. *ACM Transactions on Internet Technology*, 2(4):276–306, November 2002.

[8] J. I. Hong and J. A. Landay. An architecture for privacy-sensitive ubiquitous computing. In *The second Internation Conference on Mobile Systems, Applications and Services (MobiSys'04)*, pages 177–189, Bonston, Massachusetts, USA, 2004. ACM Press.

[9] E. Kaasinen. User needs for location-aware mobile services. *Personal Ubiquitous Computing*, 7(1):70–79, 2003.

[10] S. Lederer, A. K. Dey, and J. Mankoff. A conceptual model and a metaphor of everyday privacy in ubiquitous computing environments. Technical Report UCB/CSD-2-1188, University of California, Berkeley, June 2002.

[11] S. Lederer, J. Mankoff, and A. K. Dey. Who wants to know what when? privacy preference determinants in ubiquitous computing. In *Short Talk in the Extended Abstracts of CHI 2003*, pages 724–725, Fort Lauderdale, Florida, USA, 2003. ACM Press.

[12] H. Lei, D. M. Sow, J. S. Davis, G. Banavar, and M. R. Ebling. The design and applications of a context service. *SIGMOBILE Mobile Computing and Communications Review*, 6(4):45–55, 2002.

[13] G. Roussos and T. Moussouri. Consumer perceptions of privacy, security and trust in ubiquitous commerce. *Personal and Ubiquitous Computing*, 8(6):416–429, November 2004.

[14] W3C. Xml key management specification. http://www.w3.org/TR/xkms/, March 2001.

[15] W3C. Platform for privacy preferences (p3p) project. http://www.w3.org/TR/P3P/, April 2002.

[16] W3C. Xml encryption syntax and processing. http://www.w3.org/TR/xmlenc-core/, December 2002.

[17] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems,*, 10(1):91–102, Jan. 1992.

[18] R. Want, B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis, and M. Weiser. An overview of the parctab ubiquitous computing experiment. *IEEE Personal Communications*, 2(6):28–43, December 1995.

[19] C. Xu, S. Cheung, C. Lo, K. Leung, and J. Wei. Cabot: On the ontology for the middleware support of context-aware pervasive applications. In *IFIP NPC Workshop on Building Intelligent Sensor Networks (BISON 2004)*, pages 568–575, Wuhan, P.R. China, October 2004.