Improving Security for Ubiquitous Campus Applications

Kenta Matsumiya*  Shuhei Tamaru*  Genta Suzuki*
Jin Nakazawa†  Kazunori Takashio*  Hideyuki Tokuda*‡

*Graduate School of Media and Governance
† Keio Research Institute at SFC
‡ Faculty of Environmental Information, Keio University
5322, Endo, Fujisawa, Kanagawa 252-8520, Japan
{kenta, syu-he-, genta, jin, kaz, hxt}@ht.sfc.keio.ac.jp

Abstract

This paper identifies security issues posed by ubiquitous computing applications used in university campuses, and propose software architectures to address the issues. Applications in ubiquitous computing environment exploit interactions between personal and public devices, and adapt to user’s context. Security issues posed by these applications are privacy, usability, and hybrid scheme. Privacy Profile Negotiation Protocol (PPNP) allows users to change the granularity of their personal profile presented to profile-aware services, in order to preserve privacy, and Zero-stop Authentication System (ZSAS) provides real-time automatic authentication of users to leverage usability of user authentication in the physical space. We also present several applications to outline the usage of these systems.

1 Introduction

University campuses are favorable area to start deploying and test ubiquitous computing[6] technologies. They often have public devices, for example in computer class rooms, in libraries, and in laboratories, and personal devices, such as student’s note PCs, cellular phones, and PDAs. Applications are relatively easy to test, since they have milder regulations than in business offices, and more work force than in homes.

In ubiquitous computing applications used in campuses, users’ personal devices and devices in the environment interact, and applications adapt to user’s context, such as their preference and location[3]. For example, a user’s PDA may use the display or printer in a computer classroom to output a file, or displays in the library may notify users of new books that may interest them.

Such ubiquitous campus applications poses new security issues. We identify following three issues in this paper.

- Privacy
- Usability
- Hybrid Scheme

Privacy is an issue, because users use numerous public devices. They can be either explicitly used, with the user using input devices on the public device, or implicitly used in applications in which user’s personal devices interact with the public devices. Usability needs to be taken into account when users begin to use numerous devices, and conducting security procedure on each of them becomes a burden. Hybrid scheme would be needed when there are multiple methods for security, such as passwords, identification cards, and biometrics for authentication.

This paper proposes two software architectures which address these issues. Privacy Profile Negotiation Protocol addresses privacy issue by modifying the granularity of personal information provided to public devices, for services that adapt to such information. Zero-stop Authentication System addresses usability issue by providing interfaces for sensor-based, proactive authentication. It realizes feasibility check of automatic authentication without halting the users, using user mobility models, and also binds object to users.

The rest of the paper is organized as follows. Section 2 discusses the security issues in detail, and Section 3 explains the software architectures. Section 4 describes the application which are built using the software architectures, and Section 5 concludes the paper.
2 Security Issues

In this section, we outline each issue, and explain how they are related to ubiquitous campus applications.

2.1 Privacy

In ubiquitous computing environment, users use many of the devices pervasively available, some of which are public devices, not maintained by users themselves. Users’ personal information such as name, age, and address may be transferred to these public devices either explicitly by user input, or implicitly by device interaction. We assume that users have their personal information stored in their personal device, and that public devices may use them to provide personalized services. There are two points where personal information can be protected. Prior to transfer in the output level, and after transfer in the public device level.

First point does not effect explicit transfer, since users can always choose what to transfer. For implicit transfer, we can make users select which information to transfer each time they use an application. However, this approach may be costly in terms of burden on the users. Another approach is to create a filter for the personal information, and to restrict the information transfer according to the credibility of the public device. In this case, users’ burden is decreased, but filtering scheme needs to be created, and devices need to be categorized based on their credibility.

Second point is protecting personal information in the public device. Uncontrolled reads or copies of personal information need to be avoided, and users should be able to remove their personal information from the public device.

2.2 Usability

High usability of security mechanisms encourage users to use them effectively. Mechanisms that are difficult to use may be incorrectly used, or may not be used at all. Current security mechanisms assume to be used within cyber space, and do not regard the physical world. However, in an environment where users use myriad devices, physical aspects largely influence the usability of a system.

One method to achieve usability is to make security process proactive. Making the environment automatically initiate the security process will decrease stress on users. Several research efforts to achieve proactive authentication have been proposed[4][2].

2.3 Hybrid Scheme

New security mechanisms are constantly being developed. For example in authentication, usage of authentication tokens such as IC cards and RFIDs, and biometrics such as finger print recognition and face recognition are starting to be deployed. In near future, these new mechanisms may be widely used in campuses.

Security systems should be able to work with multiple mechanisms, since students may prefer different mechanisms, and mechanisms should be switched according to what it is applied to. For example, passwords suite authentication for legacy workstations, and ID cards or face recognition may suite authentication in building entrances. Hybrid approach is needed also for redundancy reasons. Single method of authentication can fail due to device problem, or users’ authentication token not being available. In such cases, providing multiple methods can be important. [1] uses multiple authentication methods to login to an active space environment.

3 Software Architectures

In this section, we present two software architectures we created, which address issues explained in Section 2. Privacy Profile Negotiation Protocol (PPNP)[5] is a protocol for changing the granularity of personal information transferred to public devices. Zero-stop Authentication System (ZSAS)[4] provides a model and a system architecture to realize, sensor-based proactive authentication of users.

3.1 PPNP

PPNP is used between personal devices of users and applications that adapt their behavior to their personal information. It supports exchange of personal information in a privacy kept manner. In PPNP, users maintain their personal information in a form called personal profile. Personal profile contain typed values of personal information. Table 1 shows the types used in privacy profile and their example.

Based on the privacy profile, the behavior of an application is determined. Instructions for this behavior is called control commands. Control command is generated from service rule, which is a template for application behaviors. Concrete commands are determined from the service rule, according to the personal profile. Service rules are provided by applications. The granularity of values in the privacy

<table>
<thead>
<tr>
<th>Feature and type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic</td>
<td>Name</td>
</tr>
<tr>
<td>Time</td>
<td>Birthday</td>
</tr>
<tr>
<td>Hierarchical Strings</td>
<td>Address, Affiliation</td>
</tr>
<tr>
<td>Hierarchical Number</td>
<td>Phone Number</td>
</tr>
<tr>
<td>Atomic Limited</td>
<td>Sex, Blood Type</td>
</tr>
</tbody>
</table>
profile is modified according to user rules. The granularity is changed. Figure 1 depicts the overview of PPNP. It should not be possible to derive privacy profile from control commands, and PPNP addresses this issue by modifying privacy profile before using it to generate control commands. PPNP modifies privacy profile by changing granularity of the values contained. We call the range of the value as granularity. For example, a continuous value, such as age, which is 15, can be changed to 10-20. A hierarchical value can be changed to the value of an ancestor, such that if the value is an address that is “Mathews Ave., Urbana, USA”, it can be changed to “Urbana, USA” or “USA”. Figure 2 depicts the concept of granularity. The left graph shows the case for continuous data, and the right graph shows the case for hierarchical data.

We have two ideas to implement PPNP. One is describing service rule as a symbolic description, and the other is exploiting mobile code technologies.

**Symbolic description**

In this method, a service rule is implemented as a symbolic description, which maps rules and control commands. For example, we can define:

\[
\]

This means that "In the case of ’70s age, control command is A, and in the case of ’80s age, control command is B.” An advantage of this method is that it is easier to check whether the service rule can derive back privacy profile than the latter method. A disadvantage of this method is that massive description of a service rule is needed and that the flexibility of the application is poor.

**Mobile code**

In this method, a service rule is implemented in an executable code, and transmitted to the user-side terminal. In this case, each service rule is defined as a conditional expression. For example, we can define:

\[
birthday = \text{getBirth}\{\text{profile}\};
\]

This means that "In the case where today is the user’s birthday month”, control command is “for birthday month”, and in the case where a user is 16-20, control command is “for young.” An advantage of this method is that it can be more powerful to apply privacy profile to services than the former way. A disadvantage of this method is that it can be less portable because of dependence on the program language. Moreover, in order to check whether the service rule can derive privacy profile or not, a library described in the specific language is needed. Therefore, it is limited for a programmer of services to make services with the specific language.

### 3.2 ZSAS

ZSAS realizes proactive authentication of users. It aims to authenticate users within a time constraint in order not to make them stop to be authenticated. It uses a model of user movement in which the user passes straight through a gate. Such mobility model can be seen in library exits and supermarket counters. It also senses, and binds objects to users, such as books and merchandises.

The model is denoted \(P/N \times Q/M\) model, where N and M represent number of users and objects respectively, and P and Q represent number of sensors to sense users and objects respectively. Currently, we are working with a model with one sensor each for detecting users and objects, thus \(1/N \times 1/M\) model.

We assume that the sensors are operated by an authentication server embedded in the gate, which we call gate server. The gate server checks the deadline, and processes authentication error operations if the deadline is missed. Figure 3 shows the environment we assume. Although coverage areas of all sensors are not circular, many RF sensors with omni-directional antennas have sensing area of a certain circular shape, and we assume that the coverage area of user-detecting, and object-detecting sensors are circles of radius \(R_{usr}\) and \(R_{obj}\) respectively.
As for user movement, we assume that a user walks straight along the collinear line of two sensors and the gate server at a constant velocity, $V$. By the time when a user reaches a processing deadline point (PDP), the gate server should finish both the authentication and the object processing. Then the server temporarily stores those results in its memory or storage. The gate server updates information about the user and objects by the time when the user passes through the gate (transaction deadline point: TDP). Users can obtain the feedback of authentication and object-binding by the gate server while they exist between PDP and TDP. The length between PDP and TDP depends on applications, since each application consumes different time required for feedback to users.

In a single user case, we assume that the user enters the coverage area of the user-detecting sensor or the object-detecting sensor at time $t = 0$. In this condition, the gate server should authenticate the user within the following given time:

$$\frac{R_{\text{usr}} - l}{V} - \alpha - \beta - AT \geq 0$$

(1)

where $l$ stands for the distance between PDP and TDP, $\alpha$ is the processing time of the user-detecting sensor to discover users, $\beta$ stands for the time to transfer a user-ID datum from the user-detecting sensor to the gate server, and $AT$ is the authentication time.

The velocity of objects can be obtained by approximating user’s velocity. This is because objects travel at the same velocity $V$, since the user carries objects. The gate server should process operations for the object within the time:

$$\frac{R_{\text{obj}} - l}{V} - \gamma - \delta - OT \geq 0$$

(2)

where the parameter $\gamma$ is the processing time of the object-detecting sensor, $\delta$ is the communication time to transfer an object-ID datum from the object-detecting sensor to the gate server, and $OT$ stands for the time taken by the gate server to process the operation for the single object.

Fig. 4 depicts the overview of the system architecture. The basic API provided by ZSAS is listed in Table 2. Applications register a callback function to ZSAS prior to user detection using ZSAS API, in order to have the result of the authentication notified to the application. Binding option can be specified to request ZSAS to automatically bind objects to the authenticated users. Applications also specify a user database used for authentication. Users’ IDs are confirmed with the database. Users are detected by the user sensor, and objects by object sensor separately. When the user is detected, a user’s ID is transferred to ZSAS and the authentication process is triggered. After user detection, the feasibility of Zero-stop Authentication is calculated by the model module. If Zero-stop Authentication is not possible, ZSAS issues an error message. Otherwise, the system proceeds to authenticate the user, and the user ID is confirmed in the user database specified by the application. If the authentication succeeds, and object binding is required, the binding module contacts the detection module to acquire object IDs of objects in the sensing area. Objects are constantly sensed, and the detection module keeps track of which objects are within the sensing area. The result of authentication and object binding is notified to the application as an event. The application may query object database to obtain information on the objects.

### 4 Applications

In this section, we describe two applications that we constructed using software architectures explained in Section 3. Personalized Public Message-board is an public message application that adapts to user’s preference. Secure printing is an application to print users’ documents securely, without documents being overlooked by others.

#### 4.1 Personalized Public Message-board

Personalized Public Message-board (PPM) is an public message display that can change its content of advertisements according to user’s privacy profile. PPM can be deployed in public spaces, such as entrances of buildings, li-

---

### Table 2. ZSAS API

<table>
<thead>
<tr>
<th>ZSAS API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addZSAListener(listen)</td>
<td>registers a callback function</td>
</tr>
<tr>
<td>ZSASystem getInstance()</td>
<td>returns a ZSAS instance</td>
</tr>
<tr>
<td>setBind()</td>
<td>sets the bind option</td>
</tr>
<tr>
<td>objectId bind(userId)</td>
<td>bind object IDs to the user ID</td>
</tr>
</tbody>
</table>

---

### Figure 3. Environment of The Zero-Stop Authentication System
Figure 4. System Architecture: Detection by user sensor triggers the authentication process. Application registers a callback function to the system prior to detection. Objects are continuously detected, and their ID is sent to the authentication module during object binding. Application operates actuators such as an alarm or a gate when an error occurs in the model check phase or the authentication phase.

Figure 5. User Interface of PPM

In the current version, PPM changes its contents when a user enters within a sensing area of an RFID reader. And we use IPAQ with familiar Linux to store privacy profile.

We employed ContactXML [1] on a description of privacy profile, which is XML-based. PPM is a realization of service applying users’ privacy profile as protecting privacy in public spaces.

4.2 Secure Printing

There is a physical privacy problem with current networked printers. Documents are printed with FIFO discipline, so when users print their documents, they may be overlooked by others between the time when the document is printed, and the time when they are picked up. The printer in the Keio University campus require 2 seconds to print a page at the minimum, and between 3.5 and 4 pages are printed per one queue. Therefore, if a user issues 4 pages of a document to be printed on a printer from a desktop computer far from the printer, all of his documents could be overlooked or be removed unless he arrives on the printer in eight seconds. In addition, we found that there are a lot of documents which are left or misplaced on printers. Using ZSAS, Secure Printing detects users approaching networked printers, and only prints documents of those coming close to the printer. Unlike existing printing systems, it prevents others from overlooking the documents.

Secure Printing assumes that the users carry a device such as a PDA as a client device. It uses these devices to detect and authenticate the users. Secure Printing does not use the binding option of ZSAS. It only uses the model check
Figure 7. Authentication device and screen dump of the enquiry interface

and authentication functionalities of ZSAS. Fig 6 shows the interactions in Secure Printing. A user transfers a document to a data server using a secure print command. Secure Printing uses the signal strength between the wireless network card on the PDA and a wireless access point (AP) to detect the proximity of users to printers. It assumes that an AP is placed with the printer. The AP and the printer is connected with a server, and when the user approaches AP, the server acquires user ID from AP, and authenticates the user. If the authentication is successful, the server checks the data server if there is a document to be printed. If there is, an enquiry message is shown on the client device as shown in Fig. 7. When the user confirms to the enquiry message, the document is printed on the printer.

5 Conclusion

In this paper, we have identified security issues for ubiquitous applications in the university campus environment, and proposed software architectures to address the issues. Security issues that arise in the environment where users use large number of personal and public devices, are privacy, usability, and hybrid scheme. Two proposed software architectures, PPNP and ZSAS, address privacy and usability respectively. PPNP addresses privacy by controlling the granularity with which user’s personal information is transferred to applications. Continuous data such as age can be transformed to represent a range of value instead of the actual value, and hierarchical data such as address can be transformed to represent higher level value in the hierarchy instead of the actual value. ZSAS realizes usability by using sensors and user mobility model to provide real-time proactive authentication and object binding. Time constraint is calculated from users’ velocity, radius of sensing areas, and processing time required for authentication, and error handling operation is processed if the deadline can not be met. We have created applications for each system, Personal Public Message-board and Secure Printing.

Acknowledgement

We thank the members of the Tokuda Laboratory in Keio University for their advice and support on this work.

References