A Zero-Stop Authentication System for Sensor-based Embedded Real-Time Applications

Kenta Matsumiya*  Soko Aoki†  Masana Murase‡  Hideyuki Tokuda§

*Graduate School of Media and Governance, Keio University, Fujisawa Kanagawa 252-8520 JAPAN
†Sony Corporation, Shinagawa Tokyo 141-0001 JAPAN
‡Faculty of Environmental Information, Keio University, Fujisawa Kanagawa 252-8520 JAPAN
§Tokyo Research Laboratory IBM Japan, Yamato Kanagawa 242-8502 JAPAN

Abstract—This paper proposes “Zero-stop Authentication” model and system, which realizes automatic, real-time authentication in the physical world. Applications in the physical environment such as those at library gates and supermarket counters could benefit from automatic authentication of users. These applications need to detect users using embedded sensors, and authenticate them, and bind objects to them, in real-time. To accomplish such real-time user authentication, model of user mobility, and methods to bind objects to users are required. This paper models and formulates the user mobility and time constraints, and proposes three techniques to correctly bind objects to users. A Zero-stop Authentication System built based on this model is also proposed, which automatically detects and authenticates users, and relates objects to them. The feasibility of real-time authentication is estimated using the model, and error is issued if the check fails. We also describe prototype implementation of the system, and two applications. Secure Library System uses ZSAS to authenticate users at library exits, and bind books to users. Secure Printing uses ZSAS to detect users approaching a printer, and to authenticates them, in order to print documents only when their owner approaches the printer. It protects private documents from being overlooked or removed.

Keywords

I. Introduction

Computing environment surrounding us is becoming pervasive and ubiquitous [1], with number of mobile devices and various appliances increasing. With the use of these devices, we can access computational resources with increased mobility. Moreover, sensor technologies make such an environment smart, and enable proactive behavior of applications. The applications proactively take the first action for the users by achieving both users’ and objects’ context. An example of the proactive behavior can be found in an automatic door. An IrDA sensor on top of the door detects a user coming, and opens the door without receiving any explicit commands from the user. More of such applications are starting to be commercially available [6] [7].

Turning to our daily activities, situations in which we need to authenticate ourselves are increasing. For example, we log on to computers, show ID cards when entering authorized buildings and spaces, show credit cards to purchase merchandises (on-line as well as off-line), and so on. Making environments proactively authenticate users can reduce users’ intentional interactions with the applications greatly.

The key challenge is to authenticate moving users without making them halt with security levels of authentication kept. This paper proposes a sensor-based automatic authentication; zero-stop authentication. We define “zero-stop” property as a property of an authentication system not to make moving users pause during authentication process. To achieve this property, a system needs to meet the following three functionalities:

- Correctly detecting users and objects.
- Providing active authentication that requires no input from users.
- Providing real time response.

To provide real time response, first user and object mobility need to be modeled. Modeling free mobility in which users and objects are assumed to move around through arbitrary physical point and in arbitrary direction can be difficult. Rather, we have modeled one specific class of mobility in which users and objects pass through a certain physical point in one direction. We can see such mobility pattern in real world, for example at gates in public buildings and in front of service counters. We expect that many applications can benefit if such a mobility class is formalized. Our model is called “Zero-stop Authentication”. In this model, the aim is not so much as to provide hard real-time guarantees for the authentication. Rather, our aim is to provide formulas to check if the entire authentication process can be completed within certain time, so necessary compensation can be provided. To keep the authentication safe, we adopt existing security technologies such as secure socket layer (SSL) [2] and IPSec [3]. Encryption technologies like DES [4] and RSA [5] also secure authentication process.

The rest of this paper is structured as follows. In Section II, we compare related work with our system, and discuss whether current work can achieve the requirements of the Zero-stop Authentication. Section III introduces “Zero-stop Authentication model”, and formalizes users’ and objects’ mobility in terms of soft real-time operations. Section IV explains the design of a system architecture to provide sensor-based Zero-stop Authentication. In Section V, we describe the prototype implementation of our system. Section VI explains
two applications: Secure Library System and Secure Printing. We summarize this paper, and discuss future work in Section VII.

II. Related Work

The ActiveBadge system [8] and BAT system [9] are sensor management systems for context-aware applications which tracks users and objects. In this tracking system, the users and various objects are tagged with the wireless transmitters, and their location is stored in a database. Therefore, the main goal of the BAT system is detection of users’ and objects’ accurate location. Since the objective of Zero-stop Authentication system is to build an authentication system on top of a sensor system, the ActiveBadge and the BAT can complement our system.

Intelligent Transport System (ITS) [10], especially, the electronic toll collection (ETC) system [11] allows cars to go through the toll gate without stopping. To realize non-stop payment at the toll gate, automotive vehicles are equipped with devices capable of wireless communication with the toll gate. When these vehicles enter the communication area that a toll gate covers, the toll gate begins to authenticate vehicles, and then withdraws money from banks. In this authentication process, it is necessary to identify automotive vehicles or IDs such as a credit card number or a unique number bound to a credit card number. [12] proposes the method to identify automotive vehicles by using a retroreflective optical scanner, whereas [13] identifies moving vehicles by smart cards with radio frequency (RF) or infrared (IR) transponders or RF smart tags. However, the ETC model does not address the binding problem since it assumes all the vehicles are serialized. Our model, on the other hand, deals with cases where multiple users bringing multiple objects need to be authenticated at a time.

Zero-Interaction Authentication (ZIA) [14] is an authentication system in which a user wears a small authentication token that communicates with a laptop computer over a short-range wireless link. Whenever the laptop needs decryption authority, the laptop acquires the decryption authority from the token and authority is retained only as long as it’s necessary. ZIA is similar to our model in its goal of authenticating the user without stopping them. The main differences between these two models are that our model authenticate both users and objects, and formalizes their mobility by considering the real time aspect.

Recently, there have been several research projects working on authentication in smart environments. In Gaia[15], flexible and convenient methods for authentication is provided[16]. It supports different authentication methods such as identification badges, smart accessories, PDAs, traditional passwords and biometrics. When using wearable devices as authentication methods, the environment actively starts authentication process, and authenticates users for multiple services in the environment with a system similar to Kerberos. It is close to our system in that it uses wearable devices as one of the authentication methods, and that the environment actively authenticates users. However, it does not address how to bind objects to users, and also does not consider real-time response.

III. Models for Zero-stop Authentication

The main challenge of this paper is achieving a sensor based real-time authentication which authenticates multiple users passing by an authentication gate carrying multiple objects. In realizing the authentication system, there are mainly two sub challenges: modeling user and object mobility and object binding.

To support the user’s continuous mobility during the authentication process, the authentication system needs to finish its tasks within a certain time. The necessary time for authentication depends on the hardware and software performance. Therefore we need to formalize the mobility of users and objects and utilize this formulation when designing and installing the zero-stop authentication system. Also, when multiple users carrying multiple objects go through the authentication area at the same time, the authentication system needs to distinguish which objects are whose. If the system fails, some objects might be wrongly assigned to other users.

To realize zero-stop operations of authentication, an authentication server embedded in a gate (gate server) detects users and objects by cooperating with sensors, and then authenticates users within real time. In our procedural assumption, the gate server can not process the authentication operations concurrently, because it runs according to the challenge-response manner. Moreover, we assume that a task deadline is a soft deadline. The gate server checks this deadline, and it processes authentication error operations, if a deadline miss occurs.

This paper discusses the following case: a user-detecting sensor observes $N$ users, and an object-detecting sensor recognizes $M_N$ objects, where $M_i$ is the number of objects carried by user $i$. The reason why we use two types of sensors is to make the system practical. It is considered that inexpensive sensors can be used to detect objects, while richer sensors that can perform authentication protocols are needed for users.

In this section, we introduce four models of zero-stop authentication. These models can be applied to several applications such as library applications and supermarket check-out applications.

(a) $1/1 \times 1/1$ model
In this model, both the user-detecting sensor and the object-detecting sensor sense the only one entity at a time.

(b) $1/1 \times 1/M$ model
In this model, the user-detecting sensor detects only one user, while the object-detecting sensor recognizes multiple objects at a time.

(c) $1/N \times 1/1$ model
In this model, the user-detecting sensor detects $N$ users, while the object-detecting sensor detects an object per user.

(d) $1/N \times 1/M$ model
In this model, a user-detecting sensor observes $N$ users, and one object-detecting sensor recognizes $M_N$ objects per user.

A. Models of Environment

Fig. 1 illustrates the environment we assume. Although coverage-shapes of all sensors are not circular, many RF
sensors with omni-directional antennas such as IEEE-802.11b standardized devices and RF-ID readers can detect objects appeared in a certain circular area. Thus, we model that the coverage areas of the user-detecting sensor and the object-detecting sensor are circles of radius $R_{user}$ and $R_{obj}$, respectively. If $R_{user} \leq R_{obj}$ is satisfied, two sensors and a gate server are placed as Fig. 1-(a) shows (each sensors are located at the gate). Fig. 1-(b) depicts the contrary case i.e., in the case of $R_{user} > R_{obj}$.

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.

### B. Time Constrained Operations

(a) $1/1 \times 1/1$ Model: 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:

\[
\frac{R_{user} - l}{V} - \alpha - \beta - AT \geq 0
\]  

(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:

\[
\frac{R_{obj} - l}{V} - \gamma - \delta - OT \geq 0
\]  

(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.

(b) $1/1 \times 1/M$ Model: The constraint of the authentication is the same inequality as formula 1, since the gate server also authenticate a single user in case (b). However, the gate server processes operations for $M$ objects. Therefore, it should satisfy the following relationship to realize that the user does not need to stop at the gate:

\[
\frac{R_{obj} - l}{V} - \sum_{j=1}^{M} \gamma_j - \sum_{j=1}^{M} \delta_j - \sum_{j=1}^{M} OT_j \geq 0
\]  

(3)

for $1 \leq j \leq M$, where $\gamma_j$ is the processing time consumed by the object-detecting sensor to discover object $j$, $\gamma_j$ represents the communication time to send the ID of object $j$ from the object-detecting sensor to the gate server, and $OT_j$ is the processing time to modify the state of object $j$. Formula 3 assumes that the object-detecting sensor can not concurrently scan multiple objects. If it is possible, the new formula becomes simpler: $\sum_{j=1}^{M} \gamma_j$ is substituted with $\gamma_{max}$ which is the greatest value of all $\gamma_j$. In addition, the communication time, $\sum_{j=1}^{M} \delta_j$, can be reduced, if object ID data can be transferred by less than $M$ packets.

(c) $1/N \times 1/1$ Model: We consider a more complex case than case (a) and (b): $N$ users pass through a gate carrying a single object for each. In the multiple users case, user $i$ enters into the coverage area of a user-detecting sensor or an object-detecting sensor at time $t_i$. In this case, the time-constrained computation for authenticating user $i$ is as follows:

\[
t_i + \frac{R_{user} - l}{V_i} - \alpha_i - \beta_i - AT_i \geq t_i
\]  

(4)

for $1 \leq i \leq N$, where $\alpha_i$ represents the time to detect user $i$, $\beta_i$ is the communication time between the user-detecting sensor and the gate server, and $AT_i$ is the time taken by the gate server to authenticate user $i$.

If $\forall V_i = \forall V_j$ ($i \neq j$) is met, or operations for each users are serialized like ATM in a bank, the gate server just authenticates users, following the first-in-first-out (FIFO) discipline; otherwise the gate server should reschedule the order of authentication operations to minimize deadline misses. To address this issue, we have two approaches. One is using the earliest-deadline-first algorithm [17] which schedules the user with the closest deadline first. According to this scheduling policy, the gate server can determine the priority of each user by calculating $D_i$ in the formula:

\[
D_i = ET_i + \frac{R_{user} - l}{V_i} - \alpha_i - \beta_i - AT_i
\]  

(5)

where $ET_i$ is the time when user $i$ enters the coverage area of the user-detecting sensor.

The other one is building least-slack-time scheduling [18] into the gate server. In this case, the slack time for authenticating user $i$ at time $t$ is $D_i - p_i - t$, where $p_i$ is the processing time to authenticate users.


(d) \(1/N \times 1/M\) Model: A model for multiple users carrying multiple objects for each is discussed here. The order to authenticate all \(N\) users can be determined by user selection algorithms. To realize Zero-stop operations, the gate server should meet the following formula to modify the state of object \(j\):

\[
\frac{R_{\text{obj}} - l}{V_i} - \sum_{j=1}^{M_i} \gamma_j - \sum_{j=1}^{M_i} \delta_j - \sum_{j=1}^{M_i} OT_j \geq 0
\]

for \(1 \leq i \leq N\) and \(1 \leq j \leq M_i\), where \(M_i\) is the number of objects that user \(i\) carries.

C. Object Binding

In both \(1/N \times 1/1\) model and \(1/N \times 1/M\) model, the authentication system needs to bind objects to users. Examples of objects are books in libraries, and merchandises in supermarkets. If these objects are appropriately bound to users, applications will be able to register, or charge them to the user. The main challenge is to correctly sense and distinguish objects belonging to a user. While mechanisms to sense an object is maturing, those to distinguish it, and to bind it to an appropriate user is not as thoroughly investigated.

We introduce three ideas in the following that can be used to effectively distinguish between objects belonging to a user from others’. In our assumption, objects are tagged with wireless identification devices, such as RF tags. We will classify these tags into two groups: Read-Only, and Read-Write.

- **guidance**

  The guidance approach is a technique to transform \(1/N \times 1/1\) model or \(1/N \times 1/M\) model to \(1/1 \times 1/1\) model. In this approach, users are physically guided, so only one user is sensed by the system at a time. This method has analogies to traditional object binding methods, such as in supermarkets. However users often queue in supermarkets, so enough gates to realize the zero-stop property is required.

- **insulation**

  We use an insulator to obstruct radio wave to or from the tags attached to the objects. The insulator will likely take the form of specialized containers, such as shopping carts. In this approach, the authentication system detects a user who exists close to the gate, and authenticates him or her. After that, the authorized user opens the container so that the objects are exposed to, or freed to give off radio waves. The identification of the objects recognized at that point is bound to the target of the authentication. Other users must not open their container during this process, because object binding misses occur.

- **marking**

  Objects have writable tags attached, and users use devices to write their IDs to those tags. When objects are sensed, these IDs are also sensed, and reported to the system, allowing it to bind the objects to the user.

Table I classifies each binding method by types of tags and required devices.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>BOUNDING METHODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Tag Type</td>
</tr>
<tr>
<td>guidance</td>
<td>RO</td>
</tr>
<tr>
<td>marking</td>
<td>RW</td>
</tr>
<tr>
<td>binding</td>
<td>RO</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>ZSAS API</th>
</tr>
</thead>
<tbody>
<tr>
<td>addZSALeister(listener)</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>

IV. System Architecture

In this section, we describe the design of the Zero-stop Authentication System (ZSAS), which automatically authenticates users and binds objects to them. ZSAS consists of four modules: an authentication module, a binding module, a model module, and a detection module. Applications register a callback function to the authentication module, in order to have authentication results notified to them. The binding module relates object IDs to user IDs. The model module checks the feasibility of Zero-stop Authentication using the model presented in Section III, and the detection module manages the user and object detecting sensors.

We assume that there is a gate in the environment, where users pass through, and a gate server placed with the gate. Sensors are embedded in the gate server, and the system itself also runs in the gate server. We also assume that users carry portable authentication devices such as PDAs, which we will call client devices. We assume that they are connected to network by wireless, and store user identification. We use client devices for user detection and user authentication.

Design considerations for ZSAS is as follows.

- **Flexibility**

  Since various sensors and various authentication methods exist, the system should adopt a modular architecture so that the sensors and authentication methods can be plugged and unplugged with ease.

- **Ease of Configuration**

  The system would be used by various applications and in various environments. Therefore, it needs to be easily configured according to the application or the environment.

ZSAS realizes flexibility by adopting a modular design for sensors and authentication methods. Ease of configuration is realized by various configuration files which specific parameters such as sensor threshold and authentication database locations.

Fig. 2 depicts the overview of the system architecture. The
Fig. 2. 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.

basic API provided by ZSAS is listed in Table II. 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.

A. Detection

When a user is detected, an event is issued. The system is event based, and uses such asynchronous events throughout. User detecting sensors also need to acquire the velocity of the users, in order to check the feasibility of zero-stop property. If this is not possible, the system uses a default value as the user’s velocity. When an object is detected, its ID is stored in the detection module. The binding module queries this ID to the detection module for object binding. When the object leaves the sensing area of the gate server, a corresponding ID is discarded.

Several types of location sensors exist, which express location in coordinates, proximity, or whether an entity is within a sensing area or not. In ZSAS, we aim to absorb the difference by defining a sensing area, and uniformly treating location as whether an entity is within the sensing area or not. Some sensors with less flexibility in adjusting the sensing area, such as RFID readers, may need to be manually configured to cover the defined sensing area.

Multiple sensors may be used for redundancy. Increasing the number of sensors allow some of them to fail or miss during detection. In such a case, sensor data with the same user ID or object ID should be aggregated, so that event or state would not be duplicated.

B. Authentication and Binding

Authentication method is pluggable in ZSAS. Applications specify the authentication method, and a URL of a user database in a configuration file. Authentication keys such as symmetric/asymmetric key and passwords are stored in the user database. ZSAS acquires user ID and an authentication token from the user detecting sensor, and confirms the specified user database. Applications also specify whether to conduct object binding or not. If it is specified, the authentication module consults the binding module after the authentication. The binding module queries the detection module for the objects to be bound to the user. It returns the IDs of these objects. In turn, the authentication module issues an event to the application, notifying the result of authentication and IDs of the objects.

Procedure to bind objects differs with the object binding scheme. For marking scheme, a user ID is specified by the binding module as a key to acquire the object IDs. Since user IDs are assumed to be marked to the objects, the detection module returns only the IDs of objects associated with the user. For other schemes, simply object IDs of those within the sensing range is returned.

C. Security Concerns

Dealing with security issues in ZSAS is also important to deploy our system. Unlike current authentication systems, ZSAS does not require users to input their passwords or names. Instead of users’ inputs, their portable terminals answer to the gate. Although it realizes the zero-stop property, it makes the system vulnerable. If the client device is stolen, his or her identity is also stolen at the same time. To protect users’ portable devices, biometrics is a usable and practical solution. When a user desires to utilize the client device, a fingerprint recognition system on the client device authenticate the user. Thus, only authorized users can utilize their portable devices.

In addition, communication between client devices and the gate should be protected because attackers can eavesdrop on
a series of the authentication process. It is useful to utilize existing secure communication protocols, such as SSL/TLS and IPsec. Since these protocols are able to encrypt data exchanged between the server and the client, authentication can be processed securely. Moreover, SSL prevents the server and client spoofing problem by using trusted third parties' certification messages.

V. PROTOTYPE IMPLEMENTATION

In this section, we describe the prototype implementation of ZSAS. We used Java (Sun JDK1.3) for most of the implementation to achieve portability, and C for architecture dependent programs such as those to interact with wireless LAN interface device drivers, and connected them using Java Native Interface. The prototype implementation maps a single application to single ZSAS instance, since managing multiple authentication databases and binding policies can complicate the system by a significant degree. Also, a single ZSAS instance is created for a single gate server, since sharing sensor devices between multiple ZSAS instances can be complex.

We adopted the mobility model shown in Fig. 1 (a) for the prototype system. $R_{user}$ is approximately 2m, and $R_{obj}$ is approximately 10m, as explained in following subsections. The sensors were embedded in a Smart Furniture, explained in the next subsection. We used Smart Furniture because it is designed to embed sensors as well as computers and displays, and is useful for implementing the gate server.

A. Smart Furniture

Fig. 3 depicts Smart Furniture [19], which is an experimental platform for applications that interact with the physical environment. We have designed and implemented the physical structure of Smart Furniture with Uchida Yoko Corporation, an office equipment manufacturer. The first prototype of Smart Furniture is a pole type, and composed of mainly three parts; base, pole, and case. The photos of each part are shown in Fig. 3. To maintain the extendibility and reconfigurability of Smart Furniture, each part is a reusable modular material, and we can decompose each part to acquire adaptability for various situation.

The base is designed as a round container, and can hold PCs and sensors. The pole has four slits for screws, to which we can attach diverse devices, such as displays. The case can be attached to the pole and hold a device inside it. We place various sensors inside the case.

Two types of sensors are equipped for our prototype, and they are wireless LAN device to detect users and an RFID tag sensor to detect objects, respectively. Hardware specification is described in Table III and Table IV. In the following, we will explain the implementation of the detection module and the authentication module.

Detection Module

In our current implementation, the detection module obtains sensor data from the wireless LAN device and the RFID tag sensor. Therefore, we developed their sensor driver programs for sending sensor data to the detection module.

The wireless LAN sensor driver program detects users’ portable devices using signal strength, and then provides the IP address of the user’s terminal with the highest level of signal strength among others detected by the sensor. Fig. 4 shows the source code to select the client device with the largest signal strength. To measure signal strength, we utilize IBSS mode of an IEEE 802.11b standardized device. The signal strength and the IP address of a client device is acquired from the device driver of the wireless LAN card.

The RFID tag sensor generates events when a tag enters or leaves its sensing area. Based on this two events, tag’s presence in the sensing area is maintained in the detection module. The tag ID is also acquired with the event, so the detection module
TABLE III
COMPUTING DEVICES USED IN PROTOTYPE IMPLEMENTATION

<table>
<thead>
<tr>
<th>item</th>
<th>iPAQ</th>
<th>ThinkPad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Client Device (PDA)</td>
<td>Gate Server (notebook PC)</td>
</tr>
<tr>
<td>CPU</td>
<td>StrongARM 206MHz</td>
<td>Intel PentiumIII 850MHz</td>
</tr>
<tr>
<td>Memory</td>
<td>64MB</td>
<td>256MB</td>
</tr>
<tr>
<td>OS</td>
<td>Familiar Linux v0.5.1</td>
<td>FreeBSD 5.0 CURRENT</td>
</tr>
<tr>
<td>Network Interface</td>
<td>802.11b</td>
<td>802.11b (IBSS-Mode)</td>
</tr>
<tr>
<td>Others</td>
<td>TFT Display</td>
<td>NA</td>
</tr>
</tbody>
</table>

TABLE IV
SENSOR DEVICES USED IN PROTOTYPE IMPLEMENTATION

<table>
<thead>
<tr>
<th>item</th>
<th>Wireless LAN</th>
<th>RFID Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Client Device (Wireless LAN)</td>
<td>Gate Server (RFID Reader)</td>
</tr>
<tr>
<td>Detection Range</td>
<td>160m(outside),50m(indoor)</td>
<td>15m(indoor)</td>
</tr>
<tr>
<td>Read Rate</td>
<td>NA</td>
<td>75 tags / second</td>
</tr>
<tr>
<td>Operating Frequency</td>
<td>2412-2484 MHz</td>
<td>303.8 MHz</td>
</tr>
</tbody>
</table>

![Fig. 5. Preventing the chattering problem: Two sensing areas are created, and users are not detected unless they enter the shaded sensing area. This reduces the probability of falsely detecting users moving around the sensing area, or continuously detecting a single user.](image1)

![Fig. 6. Serialization Scheme](image2)

is able to return these IDs when queried by the binding module for object binding.

We provide generic Java interface for both the user detecting sensor and the object detecting sensor, to facilitate future development of sensor drivers. Threshold of the signal strength sensor is defined in a configuration file in terms of decibels, and it can be adjusted to different environments by configuring the file.

One problem we found with using wireless LAN signal strength as a sensor, is that the system may falsely detect a user staying at a point where the signal strength matches the threshold, i.e. the edge of the sensing area. This may occur to a user already detected, or a user who is not passing a gate, but moving around it. We call such a problem “chattering”.

To cope with the problem, we set two thresholds, creating two sensing areas. Users are not detected unless the thresholds are hit with a corresponding order. Thus, the user in Fig. 5 (a) is detected, but another user in Fig. 5 (b) is not.

**Authentication Module**

After obtaining the IP address, the authentication module tries to communicate with the host to which the IP address is assigned, and then it starts an authentication process (simple challenge-response protocol).

Prototype implementation supports password based authentication. When a client device is detected, ZSAS obtains the user ID by accessing the client device. The communication link between users’ terminals and the gate server is protected by SSL. Then, the authentication module searches within a PostgreSQL based user database where user information (IDs, passwords, names etc.) is stored. Location, database name, and user name of the user database is specified in the configuration file, and can be switched to other databases. Other methods of authentication such as those using digital signatures need to be implemented.

**B. Basic Performance**

We have tested our system under the condition of adjusting wireless LAN -40db to detect and -50db to lose the connection.
This signal strength makes the authentication area as large as 2m in radius. The detection and authentication time necessary for our system was 599.33msec on average which is fast enough for the system to authenticate users before users passing through the authentication area. The standard deviation in our measurement result was 30.93.

We used commodity RFID sensor for object detection. The tag is active, and transmits it’s ID with 7 seconds interval. The sensing range can be adjusted by changing the antenna of the reader, between 50m and 10m. We used 10m range for the prototype implementation.

C. Serialization Scheme

Since we have utilized RFIDs which are not data writable and read only, we have adopted the guidance method described in Section III-C for the object binding. In guidance scheme, the sensing area of the object detection sensor needs to be restricted, so that it does not sense objects of other users in a queue. Since the object detecting sensor of our prototype implementation has a sensing range of 10m, the users are guided to queue up farther than 10m from the gate server.

Our library application and authentication system should deal with a concurrency access problem. When several users concurrently access the gate server at the same place, the gate server can not realize zero-stop property. Some tasks may fail and miss their deadline, because the gate server can not provide enough resources. To address this issue, the serialization scheme is introduced in our system as Fig. 6 illustrates.

VI. APPLICATIONS

In this section, we describe two applications that we constructed using ZSAS. Secure Library System is a library book checkout application assuming use in libraries of university campuses and of public. Secure printing is an application to print users’ documents securely, without documents being overlooked by others.

A. Secure Library System

Current libraries require users to identify themselves, and to explicitly checkout books. This can be a burden and loss of time for the users, and also a fair cost for the library. Instead, Secure Library System (SLS) allows automatic authentication of users, and binding of books to users.

SLS assumes that the users carry PDAs as client devices. It also assumes that the books have RFID tags attached, and there is a gate server at the exit of the library. Gate server has a display for users to confirm authentication results, and a device to stop users when authentication fails such as a bar or an alarm.

Upon authentication success, book IDs are also sent to SLS with the authentication result. These IDs belong to books that are within the sensing range at the time of authentication completion. SLS looks up the IDs in a book database, and acquires their information such as the title and the author. Then, it displays these information and the due date of the books on the display of the gate server, so that the users can confirm them.

After authenticating the user successfully, the RFID tag sensor driver program detects books to which RFID tags are attached. At the same time, the binding module binds the user and books, and provides the authentication module with the binding information. Finally, the output module indicates authentication results on the LCD screen of Smart Furniture for users so as to confirm details. Fig. 7-(b) illustrates the screen dump of graphical user interface which appears during the authentication process for the confirmation.

If the authentication fails, the object detection operation above is not processed. Instead of this operation, the feedback module produces error messages, and shows them on the LCD screen of Smart Furniture cooperating with the output module as Fig. 7-(c) shows. Furthermore, it also blocks the path of a user by closing the library gate, or setting off an alarm.

B. 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
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 and authentication functionalities of ZSAS. Fig 8 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. 9. When the user confirms to the enquiry message, the document is printed on the printer.

VII. CONCLUSION

This paper presents Zero-stop Authentication, a sensor-based authentication model and system, which support applications that interact with the physical world, and require automatic authentication. We propose “1/N x 1/M model”, a generic user mobility model in order to provide real-time authenticity. \( N \) denotes number of users sensed at a time, and \( M \) denotes number of objects sensed at a time. The model is evaluated from user velocity, sensing range, and authentication processing time. We also propose several methods to bind objects to users, to realize automatic check-out of objects such as books and merchandices in libraries and supermarkets. Zero-stop Authentication System (ZSAS) is built upon this model and the object binding method, and provide a framework for user and object detection, and automatic authentication.

The prototype of ZSAS is implemented in Java and C, and uses wireless LAN and RFID devices to detect users and objects. We have built two applications based on ZSAS. Secure Library System automates authentication in libraries, and binds books to users. It uses signal strength between users’ PDAs and a gate server as user detection sensor, and RFID as book detection sensor. Secure Printing is a networked printing system which only prints documents of users close to the printer, avoiding others overlooking the document. It uses the signal strength between users’ PDAs and a wireless access point placed close to the printer, to detect users.

We are extending the current system to cope with several problems which are not overcome. Two examples of future work are an object binding problem and a terminal theft problem.

In our prototype implementation, we adopted the guidance method for object binding. Since it can transform complicated models into 1/1 x 1/1 model, we were able to keep the system simple. However, in order to provide higher usability by not making users queue up, the system needs to support 1/N x 1/1 model or 1/N x 1/M model. To realize these models, we need to implement a more complex system, and at the same time apply other binding methods such as insulation and marking.

We have tried to simplify the current complicated authentication process without diminishing security level by using several security and encryption technologies. However, there is still a threat that a client device or a tag which a user should have would be stolen. For these problems, authentication technology for the device such as biometrics is usable.

ACKNOWLEDGEMENT

We thank the members of the Tokuda Laboratory in Keio University for their advice and support on this work. We also thank Uchida Yoko Corporation for their collaborating work on “Smart Furniture”.

REFERENCES


**Kenta Matsumiya** received his Bachelors degree in Environmental Information at Keio University in 2001, and his Masters degree in Media and Governance at Graduate School of Keio University in 2003. He is currently working toward the Ph.D. degree. His research interest include application layer networks and distributed computing.

**Soko Aoki** received his Bachelors degree in Environmental Information at Keio University in 2001, and his Masters degree in Media and Governance at Graduate School of Keio University in 2003. He currently works at Sony Corporation. He has developed a middleware for autonomous operation of networked appliances. His research interest include autonomous systems and mobile agents.

**Masana Murase** received his Bachelors degree in Environmental Information at Keio University in 2001, and his Masters degree in Media and Governance at Graduate School of Keio University in 2003. He currently works at Tokyo Research Laboratory of IBM Japan. He has developed an application framework for coordination of devices, and sensor distribution algorithms for sensor networks. His research interest include mobile computing and sensor networks.

**Hideyuki Tokuda** received his B.S. and M.S. degrees in electrical engineering from Keio University in 1975 and 1997, respectively, and his Ph.D. degree in computer science from the University of Waterloo in 1983. He joined the School of Computer Science at Carnegie Mellon University in 1983, and is an Adjunct Associate Professor from 1994. He joined the Faculty of Environmental Information at Keio University in 1990, and is a professor since 1996. His current interests include distributed operating system and computer networks. He is a member of ACM, IEEE, IPSJ, and JSSST.