An Anonymous Context Aware Access Control Architecture For Ubiquitous Services

Shigetoshi YOKOYAMA† Eiji KAMIOKA‡, Shigeki YAMADA‡

† NTT DATA R&D Headquarters 1-21-2 Shin-kawa, Chuo-ku, Tokyo, 104-0033 Japan

‡ National Institute of Informatics 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430 Japan

E-mail: † yokoyamasgt@nttdata.co.jp, ‡ {kamioka, shigeki}@nii.ac.jp

Abstract

A new context aware access control architecture called Anonymous Context Aware Access Control Architecture (ACA^2) is proposed. The basic idea of this architecture is based on an analogy to the public telephone service. Anonymous users can use services supported by their context information through pre-registered software components called proxies. The main features of the architecture include anonymity, access suspension caused by context changes, and active context certificates with stream verification.

1. Introduction

The spread of ubiquitous computer environments is increasing the opportunities for ad-hoc use of services in diverse environments. To make ubiquitous environments even more useful for both service providers and service users under such circumstances, a mechanism is needed to enable easy and secure provision of services under current conditions even if a trust-based relationship between service provider and service user has not been formed by pre-registration or other means.

In cases where a trust-based relationship has already been established by pre-registration for services or other means, it is common to define accessible services in terms of a user identifier (ID) and user role [1]. In ubiquitous environments, however, the following problems in access control make it difficult to apply the above system as-is.

Problem 1: The relationship between entities that need services and entities that provide services are often ad-hoc in nature, which makes an access-control mechanism based on ID and role difficult to implement.

Problem 2: Because access conditions in a ubiquitous environment are always changing, an access-control mechanism that does not accommodate such changes will not be able to control user access in a useful and detailed manner.

Problem 3: In a ubiquitous environment, the number of entities that need and provide services can be enormous. This makes it easy for operation bottlenecks to occur if access control is centralized.

In this paper, we propose a new access control architecture called "An Anonymous context Aware Access Control Architecture (ACA^2)", which solve these problems. The main idea of this architecture is to separate pre-registered area and anonymous area and attach anonymous subject to pre-registered one in ad-hoc manner.

In our architecture, context is a key element which controls access privileges and because of its dynamic nature, the system monitors the context to take away the privileges when the context changes using context management platforms.

In this paper, we define context as a composed object which has three components. One of the components subscribes events and retrieves status from sensors or other information sources. The second component has conditions by which the context analyses situation from the events and status. The last one is the component that generates events according to the situation.

The rest of this paper is organized as follows. Section 2 describes requirements to the architecture when we apply the access control mechanism to ubiquitous environments. Related works in context aware access control is surveyed in Section3. The validation of our approach is explained in Section 4. In Section 5, we illustrate a process flow using a

Proceedings of the 7th International Conference on Mobile Data Management (MDM'06)
0-7695-2526-1/06 $20.00 © 2006 IEEE

typical application scenario, before presenting concluding remarks in Section 6.

2. Requirements

An access-control mechanism must have the following functions to work well in ubiquitous environments.

Ad-hoc operation: For two parties that begin to communicate with each other without having formed a trust-based relationship beforehand, access control must be based on current conditions (context) such as the state of the service user and not, for example, on pre-registration for service use.

Tracking: In addition to determining context at the time of a service request and deciding whether to permit access, it must be possible to suspend a service if context changes during service provision and the range of permitted access is exceeded.

Distributed processing: Access control and decision making must be implemented in a distributed manner instead of centralizing access functions at a single access-control server.

An access-control mechanism that incorporates the above functions can be applied to the following example:

Usage scenario (user borrows a wireless VoIP environment): Manager-x at Company-X receives a visit from Salesman-y_1 of Company-Y. During their meeting, Salesman-y_1 needs to ask Engineer-y_2 in Company-Y's technology department a question. To do this, Salesman-y_1 makes a call with his cell phone having a VoIP/Wi-Fi function from a floor that Manager-x of Company-X is also on, and talks with Engineer-y_2 using the cell phone's voice-calling function. As long as Manager-x is on that floor, Salesman-y_1 can borrow an access point and Session Initiation Protocol (SIP) server of Company-X.

This type of distributed access-control mechanism can provide services for enterprises even on a scale of 10,000 users or several times that without having to deploy large-scale servers.

3. Related Works

Aiming for an alternative to access control based on user ID and role, a form of access control that makes use of context has come to be researched. This research field, called Context Aware Access Control (CAAC), can be divided into the following three types.

(1) Extended RBAC

As an extension of Role-Based Access Control (RBAC), this type of CAAC provides more detailed control than RBAC by applying a policy that takes context into account when deciding what access privileges to grant an access entity having a certain role in a certain context. Extended RBAC can be thought of as a means of speeding up access control between a service provider and service user that have an ongoing, trust-based relationship [2][3].

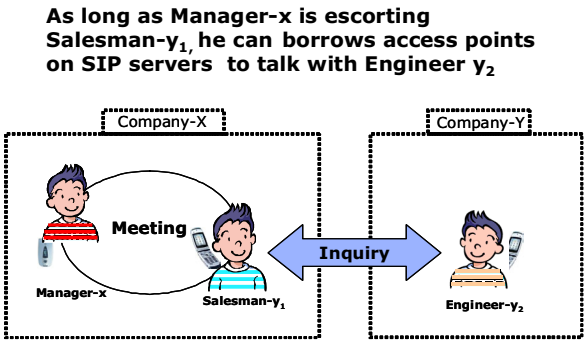

Fig.1 ACA^2 Application Scenario

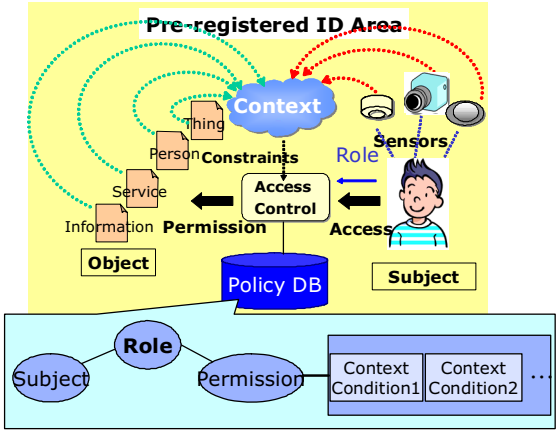

Fig.2 Extended RBAC

(2) Delegation

In this type of CAAC, a person that already has a trust-based relationship with a service provider transfers access rights to a service user. (It is assumed here that the person having that trust-based relationship with the service provider also has a trust-based relationship with that service user.) Delegation-type CAAC is built upon trust-based relationships between people and the transfer of access rights by a manual operation. This increases the burden on the person having the trust-based relationship with the service provider and makes it difficult to automate the transfer and make it more convenient [4].

flexibility in the relationship between context and permission, and assumes that the service user registers an ID beforehand as a precondition for acquiring context and that the ID and context are intertwined.

Attempting to apply the results of existing CAAC researches to service providers and users having only an ad-hoc relationship presents the following problems.

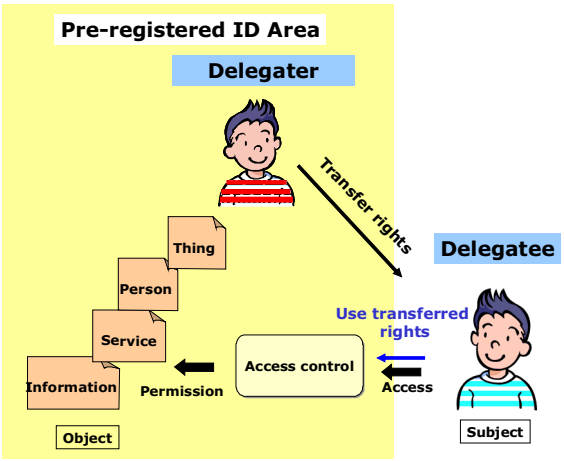

Fig. 3 Delegation

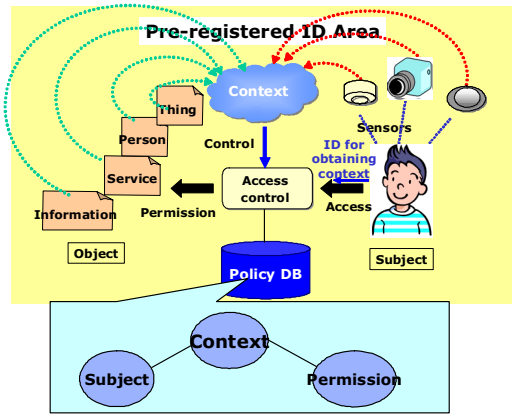

Fig.4 Pure CAAC

(1) Prior registration

An architecture that presumes pre-registration of the service user with the system is difficult to apply in a ubiquitous environment.

(2) Access control that tracks context changes

To incorporate ever-changing context into access control, it is not sufficient to merely control access rights at the time of an access request. It must also be possible to revoke access rights in the face of changing context even if access has already been granted. There is little, if any, research that discusses systems for achieving this.

Given a service provider and service user who do not have a trust-based relationship established, for example, by pre-registration of the service user, this research aims to develop a system whereby a service user can access services securely based only on the context possessed by both the service user and service provider and an access policy that describes the context/permission relationship.

Type	Pros	Cons
Extended RBAC	**Can be achieved by extending an existing and well-known solution.**	**ID and role must be registered beforehand.**
Delegation	**Can be achieved by adding a delegation function to an existing solution.**	**Places a burden on the delegatee.**
Pure CAAC	**Holds the possibility of achieving access control based only on context.**	**A new mechanism must be realized.**

Fig.5 Comparison of related works

To give some background to this development, Fig. 5 compares the related works described above.

Extended RBAC requires a user possessing a certificate to register with the system beforehand. This type of access control is therefore not suitable for a ubiquitous environment in which the relationship between a service user and service provider tend to be ad-hoc in nature. A ubiquitous environment, moreover, requires that access control accommodate diverse locations and context changes, which means that delegation-type of access control—in which someone must intervene—would not be suitable here from the viewpoints of scalability and flexibility.

4. Proposed Architecture (ACA^2)

4.1. Approach

In response to the problems, which are described in the previous section, we have decided to expand upon Pure CAAC and construct an access control system that can be used in a ubiquitous environment in which securing a trust-based relationship beforehand is difficult. This system, which we call Anonymous Context Aware Access Control Architecture (ACA^2), has three main features: (1) ad-hoc operation, (2) revocation of access rights due to context changes, and (3) context certificates based on a streaming system. Figure 6 shows a conceptual diagram of this system. In this paper, we first propose an "ACA^2 connection model" that simultaneously satisfies the requirements of ad-hoc operation and distributed processing.

We then describe an architecture for representing context validity itself as context to enable access rights to be revoked at any time because of context changes.

Next, we describe a streaming context certificate that make possible the secure revocation of access rights through context monitoring even in a network environment in which unauthorized access from the outside may be possible.

Finally, we describe total process flow using a typical usage scenario.

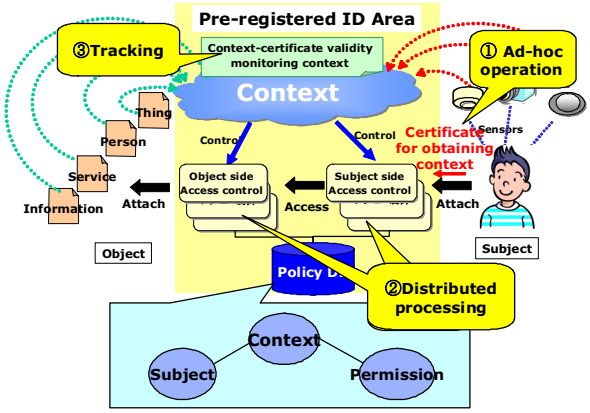

Fig.6 Anonymous Context Aware Access Control Architecture (ACA^2)

4.2. ACA^2 connection model

The use of a public telephone (pay phone) requires no pre-established contract between the user and telephone company. The user can use a public telephone in an ad-hoc manner by simply picking up the receiver. At this time, the telephone company performs access control based on the user's ability to pay the usage charge, which can be certified by the insertion of coins or a prepaid card, for example. This process achieves ad-hoc use and anonymity.

The telephone company also has a mechanism for repeatedly checking the user's ability to pay as the duration of use increases even if a payment has previously been certified and access approved. It will terminate the service if it ascertains that the user is no longer able to pay.

A similar mechanism can be applied to CAAC to realize ACA^2 based on an "ACA^2 connection model."

Figure 7 compares the public-telephone connection mechanism with this ACA^2 connection model.

We consider the following system configuration as a means of achieving a mechanism based on the public-telephone connection model. Figure 8 shows the basic architecture of this system. The system comprises the following elements:

(1) Context Servers: Group of servers providing context

(2) Proxies: Pre-registered elements providing terminals attach points to the system

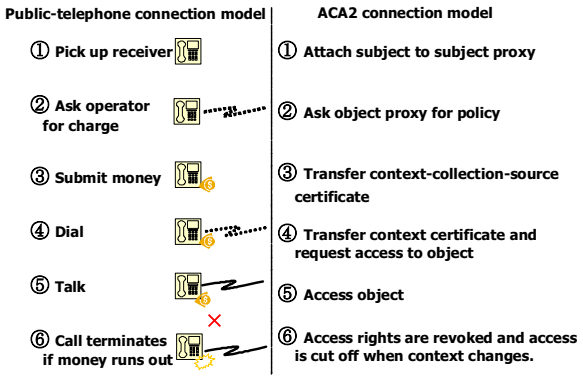

Fig.7 Public-telephone connection model versus ACA^2 connection model

(3) Terminals: Devices which subjects and objects use

(4) Subject and Object: Service consumer and service provider

(5) Sensors: Context sources

(6) Message Service: A network performing end-point authentication and providing secure-communication functions (Point-to-point messages and publish/subscribe messages) between end points (context servers, proxies and sensors)

In this architecture, Subject, who is the user of Subject_Terminal is the service user, Object, which is the user of Object Terminal is the service provider, and Sensors are generators of information that establish the current context of both Subject and Object. Here, Subject_Proxy and Object_Proxy are pooled software components prepared and registered with the system beforehand to represent the Subject and Object, respectively. They have the role of connecting an ad-hoc entity to the system.

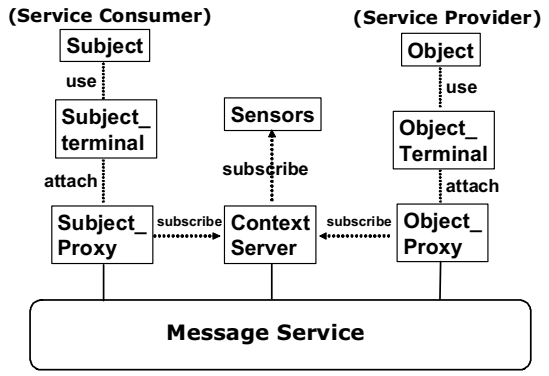

Fig.8 ACA^2 components

As shown in Fig.9, Subject_Proxy must be attached to Subject—an ad-hoc access entity—in order to connect the latter with the system. The Subject_Proxy component is pooled from within the system. On receiving an attach request from Subject, the system allocates a Subject_Proxy component in an ad-hoc manner from that pool.

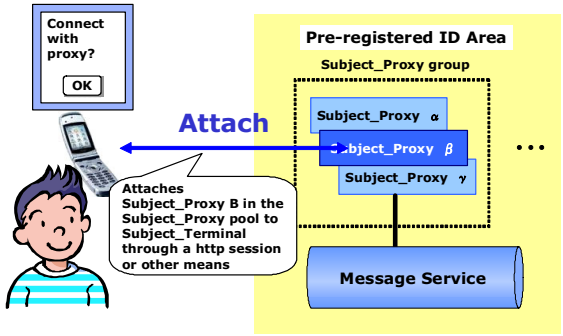

Fig.9 Context certificate and anonymity

Now, described in Fig.10, to construct a trust-based relationship, Subject sends a context-collection-source certificate (corresponding to coins in public-telephone connection) to Subject_Proxy. A context-collection-source certificate describes how Subject context is collected, and states that a reliable third party can guarantee the legitimacy of that context collection. It can also be used as a substitute for a cell phone's terminal ID or the ID of a built-in contact-less smart card, for example.

On receiving the context-collection-source certificate, Subject_Proxy prepares to collect context based on the information in that certificate.

We note here that when Subject issues an access request to Object via Subject_Proxy, the Object_Proxy representing Object notifies Subject_Proxy of the context conditions required for that access request based on access policy.

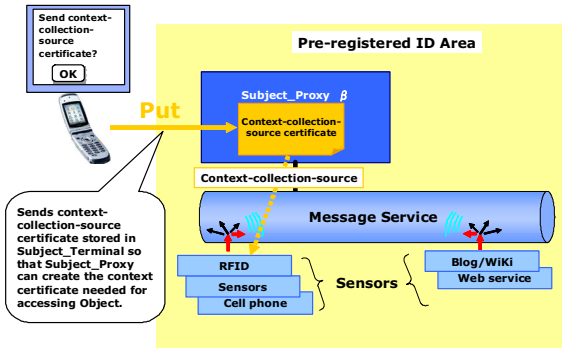

Fig.10 Context certificate and anonymity

As explained in fig.11, given those conditions, Subject_Proxy collects the context needed for access by Subject based on information from the prepared context-collection points. It groups the collected context in a "context certificate" that it signs and sends to Object_Proxy. This process enables Subject to access Object. The context certificate indicates that Subject is currently located in a certain context. Furthermore, to monitor the validityof that context at all times, Subject_Proxy subscribes to context-validity monitoring context in which change in context itself is treated as context.

4.3. Context-certificate validity monitoring context

Figure 12 shows the system for detecting change in context. To enable access rights to be revoked at any moment due to context changes, the issuing of a context certificate by Subject_Proxy will be accompanied by registration on a context server of context-certificate validity monitoring context corresponding to that context certificate. This context-certificate validity monitoring context enables context to be continually monitored for any changes. Having both the Subject_Proxy side and Object_Proxy side subscribe to this context enables changes in context—the basis for granting access rights in the first place—to be caught. If it becomes necessary to revoke already granted access rights due to context changes that have come to be detected, each of the above proxies will decide what processing to perform based on access policies.

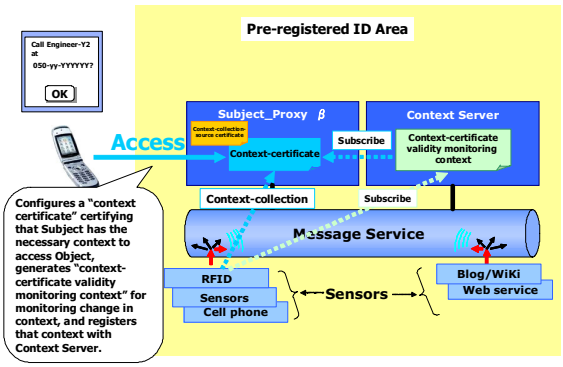

Fig.11 Context certificate and anonymity

4.4. Streaming context certificate

Given a closed and secure network, it is reasonable to assume that notification of changes in context-validity monitoring context can be conveyed from one entity to another in a secure manner. A ubiquitous environment, however, does not necessarily guarantee the use of a secure network for context subscription. As a result, a subject with malicious intent may have the capability of continuing access even if original access rights no longer exist due to context changes. This could be accomplished by blocking the network so that Subject_Proxy and Object_Proxy cannot be notified of a change in context-validity monitoring context.

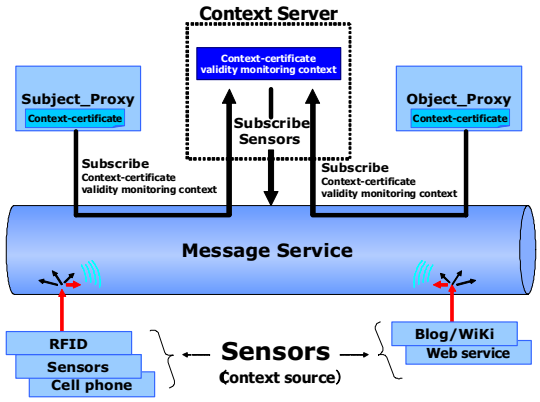

Fig.12 Context-certificate validity monitoring context

Accordingly, a robust access-control mechanism that could function effectively even in the above environment should make possible CAAC with an even higher level of reliability. We propose a "streaming certificate" to this end (Fig. 13). This mechanism delivers in a broadcast-like manner the validity of a certificate whose validity can change dynamically (as in a context certificate) in the form of streaming data.

In normal circumstances, a change in context can be found out by subscribing to context-certificate validity monitoring context on the context server corresponding to that context. Access-control processing can then be performed in accordance with that change. If, however, the subscription process itself should be blocked by intentionally cutting off the network, for example, a subscriber would not be able to learn about a context change allowing unauthorized access to be performed

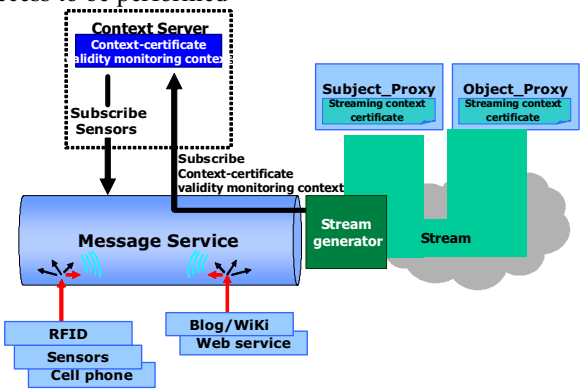

Fig.13 Streaming context certificate

To defend against this possibility, we adopt a system that continuously certifies the validity of a certificate by the use of streaming data as opposed to subscribing to context changes on an unreliable network. Here, to prevent tampering with the information flowing in the stream, the mechanism inserts the digital signature of the stream-delivery source. A subject with malicious intent is consequently unable to obstruct subscription to context-validity monitoring context by some forceful means such as cutting off the network.

5. Process flow

The process flow of ACA^2 can be broadly divided into three phases: initialization phase, access phase, and monitoring phase. Using the borrowing of a wireless VoIP environment as a typical application scenario, the following describes the execution of each of these phases.

(1) Initialization phase

(a) Registration

The Subject_Proxy group, Object_Proxy group, and context servers and sensors are registered beforehand in the system, respectively.

In our example, the Subject_Proxy group to be managed in the section where Manager-x works within Company-X is registered in the system and the Wi-Fi access point used by Manager-x within Company-X and the Object_Proxy group that manages SIP servers are registered in the system.

In addition, access points that can obtain location information on cell phones within Company-X, sensors such as active tag readers, and context servers that manage context acquired by the above are all registered beforehand in the system.

(b) Authentication

In this step, the Subject_Proxy group, Object_Proxy group, and context servers and sensors are each authenticated to the system. In our example, each end point is authenticated when activated thereby entering a logged-in state.

(c)Distribution of context-collection-source certificates

For each subject, information that certifies the context-collection source—the means of presenting context when a subject makes an access request—is distributed beforehand by trusted third parties. In our example, Salesman-y_1 receives a certificate certifying the terminal ID of his cell phone.

(2) Access phase

(a) Use and attach

Here, Subject declares use of Subject_Terminal by entering, for example, a personal identification number (PIN), and then uses Subject_Terminal to attach a Subject_Proxy within the system. Likewise, on the Object side, Object_Terminal is used to attach an Object_Proxy.

For example, Salesman-y1 can use his Wi-Fi-capable cell phone to declare use of a Subject_Proxy from the software pool of Manager-x's department in Company-X. It is assumed here that such a Subject_Proxy can be accessed from the segment that can be used freely by the Wi-Fi network in Manager-x's department.

In addition, an Object_Proxy for managing access control for a SIP server (Object) is obtained from a software pool and attached to that Object, and an Object_Proxy for managing access to VoIP-capable Wi-Fi is attached to a wireless LAN access point.

(b) Policy registration

In this step, Object_Proxy obtains an access policy for the Object in question from a policy database and loads that part of the policy that concerns itself.

We consider, for example, a person that visits a member of the section where Manager-x works in Company-X. Given that the person is a bona fide visitor to that section, the access policy in question allows the visitor to make telephone calls using the VoIP-capable wireless LAN access point and SIP server. This policy is loaded by the Object_Proxy attached to the SIP server and by the Object_Proxy attached by the wireless LAN access point.

(c) Access request

Subject sends an access request to Subject_Proxy, which then makes an access request to Object. On receiving this request, Object_Proxy requests Subject_Proxy for a context certificate corresponding to the context needed for access based on the previously obtained access policy.

For example, Salesman-y1 might search for the telephone number of Engineer-y_2 in Company-Y using his own cell phone and make a call to Engineer-y_2 via Wi-Fi. At this time, the SIP-server side in Company-X will request a context certificate from the same area that the specified employee (Manager-x) is currently in.

(d) Issuing context certificate

Subject_Proxy requests Context Server for a context certificate needed for access. The Context Server obtains sensor information needed to certify context, and at the same time, enables Subject_Proxy to subscribe to context-validity monitoring context so that it can be notified of any context changes in the future.

In our example, Subject_Proxy asks Context Server for location information on Salesman-y_1 and Manager-x while also registering context for monitoring that location information on Context Server and subscribing to that context.

(e) Obtaining access certificate

Subject_Proxy sends the context certificate needed for access to Object_Proxy, which evaluates the certificate on the basis of access policy. If Object_Proxy determines that the context certificate agrees with policy, it sends an access certificate to Subject_Terminal (Subject) via Subject_Proxy.

For example, if location information of Manager-x in the context certificate indicates that Manager-x is in Company-X, and location information of Salesman-y_1 indicates that he is situated near Manager-x, a certificate enabling VoIP use will be sent to Salesman-y_1's cell phone.

(f) Sending of access certificate and accessing object

Subject_Terminal (Subject) now sends the access certificate to Object and makes an access request to Object_Proxy. Access to Object is allowed as long as a

valid context certificate guarantees the access certificate.

In our example, this would correspond to Salesman-y_1's cell phone presenting a certificate enabling VoIP use and proceeding to use Company-X's VoIP service.

(3) Monitoring phase

Object_Proxy subscribes to context published by Content Server in order to continuously monitor the context needed for granting access permission based on information in the context certificate.

Object_Proxy cuts off a session once it realizes that a certificate required for access is no longer valid due to context changes. For example, if Manager-x should move to another floor on another matter while Salesman-y_1 is on the phone with Engineer-y_2, their call will be cut off while in progress. Even if Salesman-y_1 intentionally tries to disturb Object_Proxy to know the context change by any means, the stream context certificate mechanism can notify Object_Proxy the change.

6. Conclusion

We proposed Anonymous Context Aware Access Control Architecture (ACA^2) assuming context that does not require the user to register beforehand on the service provider side. This scheme therefore has particular value in ubiquitous environments. In addition to requiring no pre-registration, ACA^2 features continuous monitoring of context for any changes and dynamic access control performed in step with those changes.

We will implement ACA^2 on a context-awareness platform and evaluate its technical effectiveness and performance [11] and expand range of application relaxing the following limitations.

(a) The supposition that Trusted Third Party (TTP) be used to centrally manage context-collection-source certificates, context certificates, and context-validity monitoring context. The current architecture can realize anonymity to service providers but to the TTP.

(b) The supposition that context delivered by context-collection sources must be appropriate.

References

[1] David F. Ferraiolo, D. Richard Kuhn, and Ramaswamy Chandramouli, "Role-Based Access Control," Artech House Publishers, 2003.

[2] J. Canny and T. Duan, "Protecting user data in ubiquitous computing environments: Towards trustworthy environments," Privacy-Enhancing Technologies (PET) 2004, pp. 167-185, Toronto, Canada, May 2004.

[3] Tripathi, T. Ahmed, D. Kulkarni, R. Kumar, and K. Kashiramka, "Context-based secure resource access in pervasive computing environments," Proc. Second IEEE Annual Conf. on Pervasive Computing and Communications Workshops (PERCOMW04), pp.159–163, March 2004.

[4] Lalana Kagal, Tim Finin, and Anupam Joshi, "Trust-Based Security in Pervasive Computing Environments," pp.154-157, Computer Dec. 2001.

[5] A. Corradi, R. Montanari, and D. Tibaldi, "Context-based access Control management in ubiquitous environments," Proc. Third IEEE International Symposium on Network Computing and Applications, (NCA'04), pp.253–260, Aug. 2004.

[6] G. Sampemane, P. Naldurg, and R.H. Campbell, "Access control for active spaces," Proc. 18th Annual Computer Security Applications Conf. (ACSAC04), pp.343–352, Dec. 2002.

[7] J. Park and R. Sandhu, "The UCONABC usage control model," ACM Trans. Information and System Security (TISSEC), vol.7, no.1, pp.128–174, Feb. 2004.

[8] J. Park and R. Sandhu, "Towards usage control models: Beyond traditional access control," Proc. the Seventh ACM Symposium on Access Control Models and Technologies (SACMAT'02), pp.57–64, June 2002.

[9] F. Perich, J. Undercoffer, L. Kagal, A. Joshi, T. Finin, and Y. Yesha, "In reputation we believe: Query processing in mobile ad-hoc networks," Proc. First Annual International Conf. on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04), pp.326–334, Aug. 2004.

[10] Shigeki Yamada, Eiji Kamioka A, Access Control for Security and Privacy in Ubiquitous Computing Environments, IEICE TRANS. COMMUN., VOL.E88–B, NO.3 MARCH 2005.

[11] Taro Nakao, Daisuke Yamada, Tatsuya Nakamura, Shigetoshi Yokoyama, Design and Implementation of an Application-Oriented Context Awareness Framework, pp.177-184, Eurescom 2005.