Securing Personal Data in Smart Home Environments

Robbie Schaefer
Paderborn University
Fuerstenallee 11,
Paderborn, Germany

Max Ziegler
cirosec GmbH
Edisonstrasse 21,
Heilbronn, Germany

Wolfgang Mueller
Paderborn University
Fuerstenallee 11,
Paderborn, Germany

ABSTRACT

User adaptive systems, as envisioned with Ambient Intelligence (AmI), can only unveil their power, if rich information of users – including private data as their preferences and abilities as well as their usage context – can be retrieved and evaluated. However, collecting and storing this data poses severe privacy problems both legally and from a trust perspective. Therefore we propose a home automation middleware for secure management of personal profiles that allows access to profiles only for the relevant stakeholders in a specific situation.

INTRODUCTION

While the vision of ubiquitous computing [22] is already partly realized, as embedded and mobile devices are integrated in our daily lives, current research on ambient intelligence [2] tries to make these technologies truly invisible by adapting their services unobtrusively to the user's needs.

To achieve this intelligent system behaviour, much information about users, including preferences, capabilities and context information has to be collected. In addition these systems deduce information and are able to make inferences about the users behaviour. Since many wired or wireless distributed objects form such an environment, it is clear that personal data is transferred over these networks and also crosses boundaries between different kinds of networks.

Therefore it is likely that unauthorized intrusion and access to private personal data by a third party or other forms of misuse may occur to mobile devices and ambient networks. Thus, although this technology is designed to make users' lives more convenient, it also implies privacy concerns. In fact, an ISTAG study [7] showed that one major obstacle for the broader proliferation of ambient and ubiquitous technologies is the lack of trust and acceptance by the user.

To make these applications more trustworthy, we need to ensure that private data is protected from third parties. Therefore we need to guarantee that only the relevant stakeholders of a user profile are allowed (for a well defined timeframe and purpose) to access the profile in question. For this reason, we introduce an architecture that guarantees that only authorized entities (persons or applications acting on behalf of persons) are allowed to perform operations on profiles. We see an initial application domain for this architecture in smart homes, where some in-home services may not be critical regarding privacy, but access to services outside the home may require some confidentiality. Further risks (even with in-home applications) are for example with the use of wireless connections such as Bluetooth or W-LAN. Even though they can be secured, they are often not because of technical ignorance of the user.

The remainder of this paper is structured as follows. After an overview of related works, we outline the overall architecture and introduce the secured profile management, which has been implemented as a proof of concept for an ambient home environment. We close with a conclusion after having discussed several attacking scenarios.

RELATED WORKS

Since the application domains for our architecture are smart home environments, we have a particular interest in projects and studies around smart homes such as the Aware Home [8] or HomeLab [6], as they provide controllable and defined environments with a limited set of users. The focus of several related projects lies in the seamless integration of Network Protocols (IP, HomeRF, BT, GSM, ...), stationary and mobile devices (PC, PDA, Mobile Phone, Set-Top Box, ...) and different services (SMS, eMail, Radio, ...).

An enabling middleware is provided with the Open Service Gateway initiative (OSGi) [14] which defines a standard residential gateway to facilitate development and use of dynamically deployed services. The OSGi framework is platform independent and can manage broadband services as well as networks in homes, cars, and comparable environments. The idea of OSGi is to provide a software gateway that interfaces the external Internet with internal (domestic) devices and identifies requirements for interoperability and general scalability.

One of the main goals of user aware environments is that a system continously provides users with the same look and feel for their personalized services and user interfaces independent of their location (home and public). For that, a system has to provide personalization of service environment, adaptation of service environment and portability of services. A key concept there is that the technology should move into the background. This is also stressed in [3], which presents seven challenges in ubiquitous computing. Beside security issues and reliability, the zero-configuration paradigm is considered to be of utmost importance. Therefore it would be unacceptable to request user information by interactive means such as questionnaires. Rather, this data should be collected by a variety of sensors in addition to static data

which e.g. may be stored in a local user profile, or device information retrieved from public sources.

As many different information sources are used, the way this information is stored differs also quite much so that we have to take various formats for user, device and context profiles into account such as CC/PP [9], UA/Prof [23] and GUP [1].

Regarding profile management for smart homes, [19] addresses the availability of personal service collection when users are moving between terminals. They focus on filtering user profiles due to the technical limitation of the current user terminal.

Another aspect of user profiles is discussed in [18], which introduces the process of selecting relevant information from profiles and the merging of multiple user profiles. Both are required when conflicts of resources or conflicts of interests occur. The former is about sharing limited resources or services, the latter occurs when two or more users have different interests and the system has to decide which user's command it will finally execute.

[17] investigates on human aspects in smart home applications describing user's attitudes; especially how users prefer to interact with their environment. They installed several applications into their test lab and concluded that locating the user to offer situation-dependent services and the seamless integration into the users' everyday life are both essential for smart home systems.

Quite a lot of solutions for intelligent environments use data mining and other techniques to model the user. [5] for example uses inhabitants activity patterns for identification which is extended in [12] for multiple inhabitants in order to make the home user aware. Such techniques require that the collected information and the drawn inferences are not to be passed to unauthorized entities.

To this end, the AETHER framework [4], defines a security management architecture for access control and trust establishment. AETHER is based on a decentralized administration presuming a set of previously unknown pervasive entities with secure associations.

In order to secure private data, we rely on several standardized algorithms and protocols considering security aspect. Firstly, the inter node communication within our system is TLS enabled (Transport Layer Security). TLS (aka SSL Version 3.1) provides payload encryption and certificate-based authentication, which is based on X.509 certificates. Similar to the Kerberos Network Authentication Protocol, we introduce a ticketing system. Other standardized security techniques can be found in [21].

PROFILES FOR AMBIENT INTELLIGENCE

In order to establish an intelligent communication environment, an underlying system which is likely composed of distributed objects will gather quite a lot of information about the environment and the user. The more (useful) information a system has, the better is the chance to become invisible and adequately adapted. For example, if a user walks into a room and it is automatically illuminated, he/she does not have to care about the underlying technology (use the light switch). This can only apply, if the system has means to detect when a user enters a room as well as the possibility, to operate the lights.

A very flexible way to model users and their situation is to use a set of related profiles that describe all relevant aspects. As there are numerous ontologies available for this task (such as [15] or [10]) we will constrain ourselves to the description of the main profiles.

In the centre is of course the user profile which may consist of identity related information, capabilities and preferences. Identity related information may for example be a user-ID, name, gender, date of birth. And this already provides some pitfalls: While some applications may have a valid interest for example in the birthday in order to pass an age verification process, for many other applications this information is unnecessary. If such information is collected and stored just for the case that some application might have a use for it, then this can be a violation of data protection laws. For example the German "Federal Data Protection Law" points out that the avoidance and economy of data is of key importance. This means that the collection and processing of unnecessary person based data has to be avoided and if possible this data should be anonymized.

The same applies also for capabilities and preferences, as individuals may not want this data to be distributed freely, although specific applications can make use of it to make life more convenient.

Related to the user profile are device profiles from devices that are in the range of or operated by a user. Since a typical device profile contains a hard- and software description of a device, device profiles do not seem to be relevant with respect to privacy. However, dynamic parts of the profile may still convey information on the user of the device. For example the time of incoming and outgoing calls including numbers and duration are usually stored in the phone and would allow quite many inferences about the user.

Finally, the context a user is in, can be stored in temporal profiles, which means that data retrieved from different sensors or other sources is prepared to be accessed via the user profile. As a prevailing example may serve the location of a user and many location-based services which are already in operation today. Again, this location data must be used by applications which are authorized by the user only.

For the middleware, as described in the next sections, we abstract from dedicated profile formats as shown in the related works. We rather provide a uniform and generic database schema that takes the profile information and privacy rules into account (c.f. section). Therefore we assume transformations between concrete profile formats and the internal generic database profile schema which can be implemented

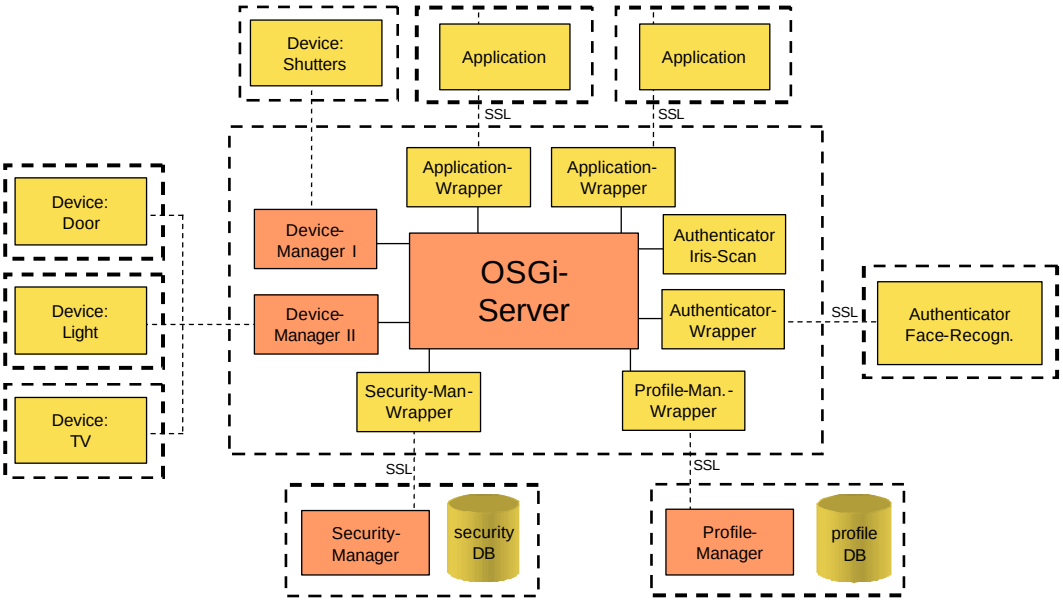

Figure 1. Architecture Overview

to support a variety of existing profile formats.

ARCHITECTURE

The main motivation of our *Secure Profile Server* system is the provision of services and secure access to user and application profiles. The system serves various controllable devices in a smart environment like a home network.

Consider the following scenario for a brief motivation. A user arrives at home and would like to open the front door. As opening the front door is a safety-critical application, authentication is required, so she has to pass a biometric authentication process. After entering, she likes to see the latest news and turns on the TV. Therefore, the TV application has to read her profile from the profile server, a list of her favourite channels is displayed, from which she selects the news channel. Since watching TV is considered to be not safety critical, a simple video tracking system is sufficient to prove her identity [16].

Presuming that a user wants to accomplish more safety critical tasks, for example, editing sensible private data or executing online banking transactions, we need some more security than before. That is where the *security levels* come into play. Different security levels for different services or for specific user profile entries (e.g. bank data) allow us to manage access to sensible or public data. To log on with a certain security level, a user has to pass different authentication tests. For a higher security level, this usually means more security by the application of more authentication methods than for a lower one. Where a single password is sufficient to protect a user's personal email account, she has to pass an additional iris scan or a face and fingerprint scan to access her bank account and execute transactions from her home environment.

Architecture Overview

As the complete system is designed to be open and extensible, a distributed architecture was chosen. Communication may use TCP/IP over commonly available technologies like Wireless LAN (IEEE 802.11) and Bluetooth (IEEE 802.15) or power line systems as the European Installation Bus (EIB) [11]. To guarantee security in distributed system, a SSL / TLS based communication with X.509 certificates is used [21]. To make sure that there are no applications that can spoof their identity to the Security Manager or Profile Manager, mutual authentication between communication partners is required. A challenge-response protocol verifies a component's identity based on its public keys.

A typical setup of the system is shown in Figure 1. The OSGi instance, where all components have to register, runs on the main server (in the center). In summary, the system has five main components interacting with a set of applications, which are outlined in the following paragraphs.

Applications. Applications offer services to users and run, e.g. on a PC (accessed, for example, over a touch screen display or a speech recognition system), a PDA or mobile phones. On request, applications turn to a Device Manager to control devices or to a Profile Manager to access user profiles, e.g., a user's preferred music or personal data.

Security Manager. To guarantee security through the complete system, we employ a Security Manager and a ticket system that is similar to Kerberos. To access a service, an application has to sign up for a ticket, which transfers certain access rights to an application on behalf of a user. This means, that the user implicitly transfers her access rights to the application, which in turn executes the request for the user.

A ticket is signed by the Security Manager using a DSA signature scheme to prevent forgery or tampering. The ticket is only created if the requesting application resp. user has sufficient access rights, which are stored in a separate security database in form of access control lists [20].

Profile Manager. All profile data are held in a profile database, which is only accessible via the Profile Manager. The profile manager permits to read or write data only if the requesting application can prove sufficient access rights with a ticket. To enhance security for sensible data, profile entries in the database are encrypted by the Profile Manager.

Device Manager. The Device Manager monitors all services that are provided by *devices* within the environment, e.g., doors, lights, tv, cd-players. An example for a *service* could be 'Open the front door', 'Switch the tv to channel 5' or 'Set light intensity to brightest'. Devices are configured to trust only a certain Device Manager, identified by its public key or a password (for low-security devices like a lamp). Like the Profile Manager, the Device Manager only grants access to the service after receiving a valid ticket with sufficient access rights from the requesting application. If necessary, a value is returned to the application.

Authenticators. Within the system, there exist one or more authentication points in the environment (house), the authenticators that provide authentication services for various security levels. For example, there can be a simple password authenticator to prove an identity by password or face recognition for higher security. These authenticators can all be available at the same time and can be dynamically chosen for certain authentication tasks.

OSGi Server. The main server is based on the *Open Platform Gateway Interface (OSGi)* [14], a standard that allows to dynamically add, remove, start or stop plugins, called *bundles*, on-the-fly. In our implementation, we have chosen the Oscar server [13]. All components of the system have to sign up at the OSGi server to provide or use any services.

Security Levels and Access Control Lists

Our system implements different security levels, each demanding a certain amount of authentication points. Assume, we have five different authenticators available in our environment and consider Table 1. Each of the authenticators provides a different level of accuracy and security, depending on the method of authentication.

Authentication Method	Points
Password	25
Smartcard and PIN	45
Face Recognition	70
Fingerprint	80
Retina-Scan	100

Table 1. Sample Authentication Methods

We can now define the security levels where each demands one or more authenticators. For example, if a medium security level demands 50 points, a user can either authenticate by a password in combination with a smartcard / PIN (25+45 points) or by means of any higher authentication method (70, 80, 100 points). Once a user is authenticated for a specific security level, she has a set of access rights (*accessMode = R, W, X*) for a profile entry (*attribute*):

$$userID \times secLevel \times profileID \times attribute \mapsto accessMode \quad (1)$$

Thhe former expression means, if the user is authenticated using a high security level, she will have more access rights than in a lower security level.

By default, all communication between the system components is SSL / TLS enabled, and the components perform mutual authentication. However, some security levels may require stronger encryption, which may be implemented by choosing a special cipher algorithm or by increasing the cipher keysize. Note, that it is also possible to go without any encryption or component-authentication at all, for example, to switch lights on or off at the lowest security level. Following this, there is complex computation involved for encryption and authentication, which might lead to performance problems when working with low performance devices like PDAs or even smartcards.

Ticket-Based Authorization

As mentioned earlier, access to services is granted only with a valid ticket. Tickets are created and signed by the Security Manager on request, after the user authentication for the corresponding security level has taken place. A ticket includes all information needed by the Profile Manager or Device Manager to decide whether access (read, write or execute) to a service can be granted or not. In summary, a ticket comprises the following entries:

- Application's identifier – for debugging / logging access events
- Application's public key – the application (the ticket owner) can be identified by its public key, so that ticket replay attacks can be avoided.
- Issuer's public key – validates the ticket and identifies the issuer can be identified. A Profile Manager or a Device Manager then decides whether it trusts tickets that have been signed by the respective issuer.
- Access control list – gives the access rights to a profile element, derived by equation (1). Note that the access rights are user-, profile- and security-level - specific, and that the application (as the ticket owner) receives the user's access rights.
- Timestamp / Lifetime – gives the date and time when the ticket was created and the duration of its validity.
- Signature – the whole ticket is hashed and the resulting hash value is then signed by the Security Manager to prevent forgery and tampering.

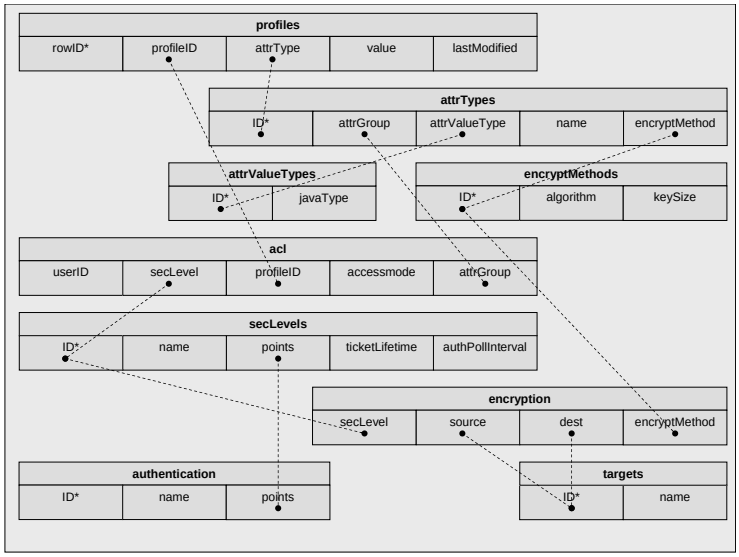

Figure 2. Generic Profile Database Scheme

Generic Database Scheme

To overcome problems – which occur through the support of several different profiles and formats – we designed a flexible data schema that also incorporates security features, allows distribution amongst different databases and implicitly obfuscates user data. Figure 2 visualizes this schema. For simplicity, we provided the security related items and profile data in the same scheme although we separated them in the architecture (c.f figure 1).

A profile table consist of a key, a profile ID which again may serve as a pointer to the access control list, an attribute type and value and a timestamp. The attribute type however is not a fixed value but rather a pointer to another table, which amongst others determines the encryption of the value. So instead of having tables with a clear and predefined structure, the result is a table with several numbers and encrypted values which – as a side effect – are very hard to make sense of. Through the distributed nature of the schema, it can be difficult even for people who are allowed to administer parts of the database to read private data, as the encryption mode may be found in a different place.

The only way, to obtain data for a service or a user is via the profile manager. To this end, the part of the scheme that belongs to the security database provides access control lists that apply to certain profiles, attribute groups and user Ids.

Example

To outline the previously introduced system, we give a simple example requesting a profile entry. Access to other services like writing profile data or executing a service (opening a door, playing music, watching TV) is similar to this example. For our example, presume that a user wants to read her bank account statements as given in Figure 3. Note that, due to the required security level, different encryptions and authentications may apply to the different communication channels.

1. The user first opens an application on her PDA and asks for her bank account statements.

2. When the application does not have valid tickets for the current user and the requested profile entry, it contacts the Security Manager (which resides on the OSGi server) for a ticket. In this case, we have $profileID = userID$, when the user accesses her own profile.

3. The Security Manager checks, which security level is required for read access for the profileID and userID.

4. Depending on the required security level, a trusted authenticator or a combination of authenticators is chosen by the Security Manager, and the user is asked to authenticate. As the user implicitly transfers her access rights to the requesting application, the application name and security level are presented to the user to avoid misuse.

5. If the authentication is successful, the Security Manager creates a ticket that allows the requesting application to place service requests on behalf of the user.

6. Having a proper ticket to access the desired profile data, the application asks the Profile Manager for the profile entries containing the bank account statements.

7. The Profile Manager checks the validity of the ticket, following 5 steps:

   - Is the ticket still valid (lifetime entry), or is it not yet valid (timestamp)?

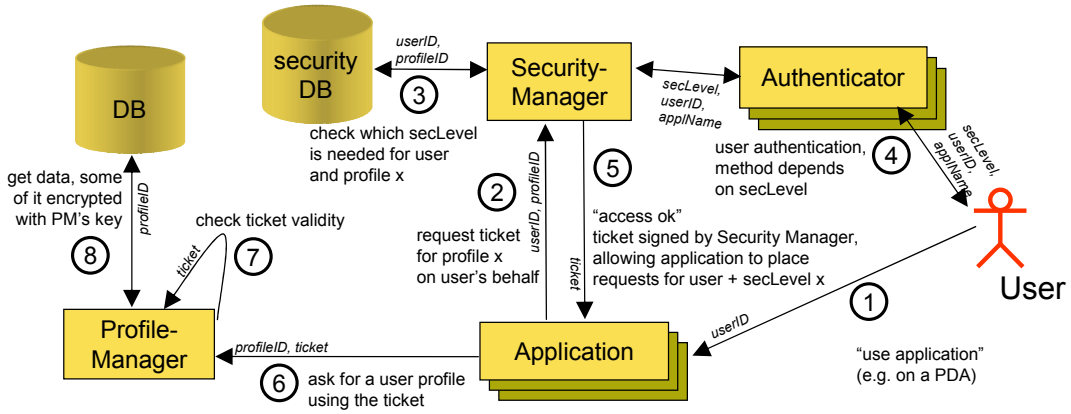

Figure 3. Example: Requesting a Profile Entry

- Is the ticket issuer a trusted Security Manager? (identified by the issuer's public key)
- Is the ticket signature correct? This can be validated by the issuer's public key.
- Is the requesting application the legitimate owner of the ticket? (identified by the application's public key, which was used in the application-authentication)
- Are the application's rights given in the access control list sufficient to access the profile entry?

8. If the previous step is successful, the Profile Manager reads the required profile entries representing the bank account statements from the profile database, decrypts them (if necessary) using its secret database key and finally returns the data to the requesting application.

Potential System Attacks

Following Figure 3, we finally identify possible system attacks and give corresponding countermeasures.

Ticket Replay Attacks. In Step 6, a malicious application might try and use a stolen ticket to request a profile from the Profile Manager. By default, the component-authentication is executed as described in the Architecture. In this case, the Profile Manager immediately recognizes that the requesting application is not the legitimate owner of the ticket since the public key, which has been used within the authentication process is not identical to the one in the ticket. If there is no component-authentication, a ticket replay attack might be possible. Additionally, through the lifetime field expired tickets are rejected by default.

Forging Tickets. An application might try to create a ticket on its own to gain more access rights. That ticket will be rejected by the Profile Manager because either the signature of the hashed ticket is not correct, or – if the malicious application signed the ticket – the Profile Manager will not issue the ticket since it only accepts tickets that are signed by trusted Security Managers.

Component Spoofing. An attacker could install bundles into the system that do not behave correctly, i.e., spy out user data or passwords. For that she could fake one of the following components:

- Authenticator. This would be of great help for the attacker since she could create an authenticator that always returns a successful authentication.
- Security Manager. Faking a Security Manager would not be of any use since the tickets that are signed by the Security Manager are simply rejected.
- Profile Manager / Device Manager. If someone fakes a Profile Manager or a Device Manager, she could collect tickets of applications that request access to profiles or devices. She could then turn to the real Profile Manager and illegitimately ask for access to services.

However, these attacks are prevented by using the component-authentication.

Forging the Authenticator Answer. By intercepting the communication between authenticator and Security Manager (In Step 4 in Figure 3), an attacker might forge the return value of the authenticator to make the Security Manager believe that the user has successfully authenticated. This risk can be reduced by always using SSL / TLS encryption for the communication between authenticator and Security Manager.

Requesting a Ticket on Behalf of Other Users. A malicious application might request a ticket for a user with very high access rights, for example, an administrator. Since the user has to authenticate and agree to the request if a ticket is requested, the Security Manager would not return a ticket to the application (Step 5 in Figure 3). Another variation of that attack could be requesting a ticket for the current user without the explicit instruction of the user. This attack is also avoided by telling the user for which application she has to authenticate and whether she wants to transfer her access rights to this application.

Illegitimate Database Access. An attacker might try to get direct access to the profile database, i.e., not through the Profile Manager. However, the database itself is password-protected, and furthermore the sensible data are encrypted with a key that is only known to the Profile Manager. Be-

sides, usually the Profile Manager and the profile database run on its own machine which makes access without the Profile Manager even more difficult. Illegitimate database access on the usual way, i.e., with the Profile Manager is not possible since it will not return any profile data if the ticket is not valid. This includes forged tickets (see above) as well as tickets that are signed by untrusted Security Managers.

CONCLUSIONS AND FUTURE WORK

In order to protect personal data in smart home environments, we have introduced an architecture for a secure profile management middleware with several components: the security manager, profile manager, device manager and several authenticators. The architecture is based on the OSGi framework for smart homes. The main principle we have realized is a ticket service for secure profile and user/ application access right management. An important aspect is, that the user's rights do not only depend on the actual user, but also on their authentication methods. This way, it is guaranteed that a tradeoff between security and comfort remains possible.

The architecture as described in this paper has been implemented as a proof of concept. Despite of the complexity of the architecture, accessing home services is fairly easy for the end user. For many home services, it is only necessary to present the smart card which is quite intuitive.

For the future, we plan to integrate the previously discussed concepts in the ambient computing laboratoy (ACLAB) of our instituite. The ACLAB is designed as a living environment and allows the remote control of appliances such as lights, doors, shutters etc. as well as entertainment facilities which can be tailored to the preferences of a user. A deep integration in such an environment will enable feasability and usability tests and also allow further exploration for example with different autheticators.

Acknowledgements

The work described herein is partially funded by the IST project UBISEC (IST-2002-506-926).

REFERENCES

1. 3rd Generation Partnership Project: ”Data Description Method (DDM) - 3GPP Generic User Profile (GUP)”, Technical Specification, 2004.
2. E. Aarts: ”Ambient Intelligence in Homelab”, Royal Philips Electronics.
3. G. D. Abowd, B. Brumitt, S. A. N. Shafer: ”At Home with Ubiquitous Computing: Seven Challenges” Proc. of Ubicomp2001, LNCS, 2001.
4. P.G. Argyroudis and D. O'Mahony: ”Securing Communications in the Smart Home”, EUC2004, LNCS 3207, 2004.
5. D.J. Cook, L.B. Holder: ”Graph-Based Data Mining”, IEE Intelligent Systems, 15(2):32-41, 2000.
6. www.research.philips.com/technologies/misc/homelab
7. IST Advisory Group: ”Ambient Intelligence - from vision to reality”, draft consolidated report, 2003.
8. C. Kidd, R.J. Orr, G.D. Abowd, C.G. Atkeson, I.A. Essa, B. MacIntyre, E.D. Mynatt, T.E. Starner, W. Newstetter: ”The Aware Home: A Living Laboratory for Ubiquitous Computing Research”, CoBuild99. Pittsburgh, PA, USA, 1999. www.cc.gatech.edu/fce/ahri
9. G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. H. Butler, L. Tran: ”Composite Capabilities/Preference Profiles (CC/PP): Structure and Vocabularies 1.0”, W3C Recommendation, 2004.
10. P. Korpip, J. Mntyjrvi: ”An Ontology for Mobile Device Sensor-Based Context Awareness”, CONTEXT 2003 pp. 451-458
11. ”Konnex Device Network, Volume 3: System Specifications, Part 1: Architecture”. Konnex Association, March 2002. www.konnex.org
12. R. Mehta, D.J. Cook, L.B. Holder: ”Identifying Inhabitants of an Intelligent Environment using a Graph-Based Data Mining System”, Proc. of the Florida Artificial Intelligence Research Symposium, pp. 314-318, 2003.
13. ”Oscar - A Java Open Source Implementation of OSGi 3”, February 2005. oscar.objectweb.org/
14. The OSGi Alliance: ”The Open Services Gateway Initiative Platform - Dynamic services for networked devices (OSGi)”, March 1999. www.osgi.org/
15. D. Preuveneers, J. Vd. Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, K. D. Bosschere: ”Towards an extensible context ontology for Ambient Intelligence”, EUSAI2004, LNCS 3295, 2004.
16. A. Ranganathan, J. Al-Muhtadi, S. Chetan, R. Campbell and M. D. Mickunas: ”MiddleWhere: A Middleware for Location Awareness in Ubiquitous Computing Applications”, CS Dpt. - University of Illinois, LNCS 3231, 2004
17. K. Rentto, I. Korhonen, A. Vtnen, L. Pekkarinen, T. Tuomisto, L. Cluitmans, and R. Lappalainen: ”Users' Preferences for Ubiquitous Computing Applications at Home”, EUSAI2003, LNCS 2875, 2003.
18. B. Salem and M. Rauterberg: ”Multiple User Profile Merging (MUPE): Key Challenges for Environment Awareness”, EUSAI2004, LNCS 3295, 2004.
19. J. Saska, T. Marenic: ”Establishing Virtual Home Environment across terminals with diverse capabilities”, 7th Int. Conf. on Telecommunications, Zagreb, Croatia, 2003.
20. H. Shen and P. Dewan: ”Access Control for Collaborative Environments”, CSCW'92, ACM Press, 1992.

21. W. Stallings: ”Cryptography and network security: principles and practice”, Prentice Hall, 2002.

22. M. Weiser: ”The Computer for the 21st Century”, Scientific American 265(3): 94-104, 1991.

23. Wireless Application Protocol Forum: ”Wireless Application Group User Agent Profile Specification”, 1999.