A Hierarchical, Agent-based Approach to Security in Smart Offices*

Iván Marsá-Maestre, Enrique de la Hoz, Bernardo Alarcos, and Juan R. Velasco

Departamento de Automática, Universidad de Alcalá
Edificio Politécnico, Ctra N-II Km. 31,600
28871 Alcalá de Henares, Madrid, SPAIN
Telephone: +34 918856644, Fax: +34 918856923
{ivmarsa,enrique,bernardo,juanra}@aut.uah.es

Abstract. As electronic devices become more and more pervasively integrated in our daily routine, security concerns start to become evident. In the last years, there has been an increasing interest on the topic of security in smart environments. One of the most challenging environments regarding security are smart offices due to the high number of potential users, devices and spaces, and the diversity of security roles. This paper presents a security solution for an agent-based architecture for the smart office. This security solution is potentially applicable to generic smart environments, but it suits particularly well to the smart office scenario, taking advantage of the particular characteristics of the environment to satisfy the security requirements. The result is a hierarchical, agent-based solution, flexible and scalable enough to be applicable to different smart office scenarios, from small businesses to large organizations.

1 Introduction

One of the main lines of research in agent technology is using software agents to automate environment personalization, so that users are released from the routine tasks they should perform otherwise to change their environment to suit their preferences and to access to the available services. The goal we seek is a *smart environment*, able to adapt itself to the user needs and to provide customized interfaces to the services available at each moment. To achieve this goal, we propose to use multi-agent systems, as they have been revealed as good technology for developing distributed, autonomous, and intelligent systems.

In our previous work [1] we designed and implemented an agent-based software architecture for smart homes. We are now extending this architecture to make it applicable to other smart environments. In particular, we're interested in smart office environments, as there are significant challenges in the automatization of such environments derived from their inherent characteristics, like the high number of potential users or the diversity of potentially available services.

* This work has been supported by the Junta de Comunidades de Castilla la Mancha grant JCCM-PBC-05009-2, and by the Universidad de Alcalá grant UAH-PI-2005/082.

One of the first challenges we have had to face while shifting from smart homes to smart offices is the design of the security architecture. In smart home environments the main goals are user comfort and easy deployment of new devices, so security is usually left apart or focuses mainly in transparency and privacy enhancement. Office security, however, has more rigorous security requirements, specially if we are dealing with large organizations, where there may be hundreds, or even thousands, of employees with different access needs and security clearances. This paper analyzes those requirements and their specific differences from those of smart homes, and proposes an extension to our agent-based architecture to provide security services for the smart office.

The rest of this paper is organized as follows. Section 2 describes the most relevant aspects of the architecture of our smart office, using a typical use case to illustrate them. Section 3 summarize the key security issues for the smart office scenario. Section 4 presents our security architecture, describing the functionality of the different agents that provide the security services. The last section summarizes our main contributions and sheds light on some future research.

2 The SETH agent architecture

The security architecture proposed in this document is being deployed over our platform SETH (Smart EnvironmenT Hierarchy), which is an extension of the iHAP architecture we developed for smart homes [1]. Detailed description of the SETH architecture is beyond the scope of this paper, and can be found in [2]. In this section we outline the most relevant characteristics of the architecture as far as our security proposal is concerned.

Our architecture relies on the concept of *smart space*s: specific, self-contained locations within the environment, which may be hierarchically arranged if the specific characteristics of the environment requires so. A mandatory device in any SETH smart space is the *Smart Space Agent Platform (SSAP)*, which contains the agent platform which supports the existence of all other agents in the smart space, and hosts the higher level agents of the system.

We may find different kinds of software agents in a typical SETH smart space. The *Smart Space Coordination Agent (SSCA)* provides device, service and context discovery to all other users or agents in the smart space, and to SSCAs of other smart spaces. *Device Agents* provide a common interface to devices, so that other agents in the system may use them regardless of specific hardware issues. *System Agents*, like context and security agents, reside in the SSAP, and add an additional layer of intelligence on top of the devices in the environment through control and coordination mechanisms. *Service Agents* are intended to provide services directly to the user, and they may be *persistent*, if they are always active at a given SSAP, and *non-persistent* or *mobile*, if they they are created by the SSCA for each request of the service, move from one SSAP to another when user location changes, and are destroyed once the use of the service has concluded. Finally, *Personal Agents (PAs)* are the very representatives of

users in the environment, able to issue requests to system, service and device agents on behalf of their associated users.

Figure 1 illustrates a typical service access case in the SETH system. User *Alice* enters the *presentationRoom* smart space. Context agents notify both the PA and $SSCA_{presentationRoom}$ of this event (1). The personal agent, knowing that its user is going to make a presentation, asks $SSCA_{presentationRoom}$ for an agent providing a presentation service (2). No such agent is persistently active in the SSAP, so $SSCA_{presentationRoom}$ creates an instance of a Non-persistent Presentation Service Agent (*NPPSA*) (3) and returns its address to the *PA*(4). The personal agent then contacts the *NPPSA* and requests it to present a slideshow of a given document stored in the computer at *Alice*'s deskroom (5). Then the *NPPSA* contacts $SSCA_{deskroom}$ to learn who to ask for the document (6). He is given the address of a persistent File Transfer Service Agent (*FTSA*) (7), which the *NPPSA* contacts to request the file (8). The *FTSA* obtains the file from the device agent (*DA*) associated to *Alice*'s desktop computer (9) and transfers it to the *NPPSA* (10). Finally, the *NPPSA* requests the device agent of the projector in the presentation room to display the slideshow presentation (11).

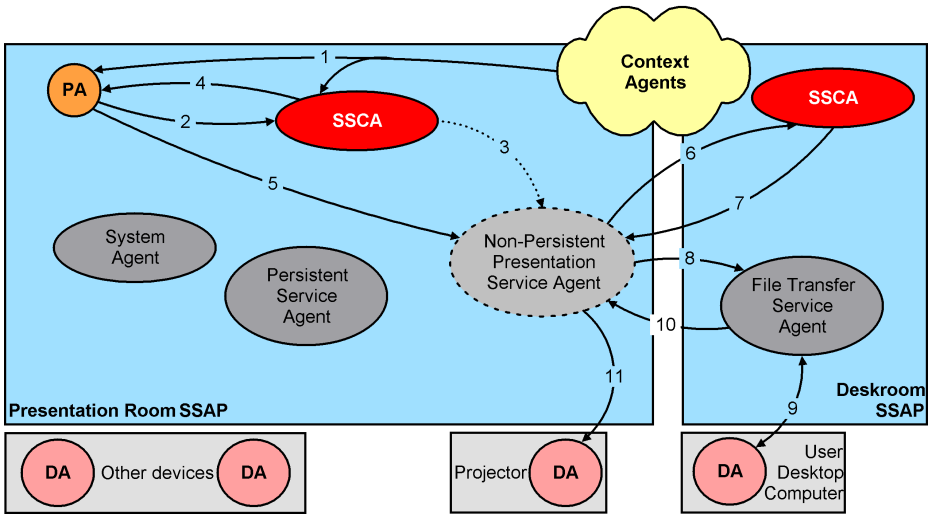

Fig. 1. Access to services

Any typical interaction such as the described above may involve service discovery requests to different SSAPs, requests to service and device agents, and even access to resources within the user's desktop computers. This flexibility of the interaction mechanism provided by agents greatly improve functionality of the smart space, but also raises some relevant security concerns that must be addressed in order to ensure there is no misuse of the provided infrastructure.

3 Security Issues for the Smart Office Scenario

From a functional point of view, security is intended to assess the risks present on a given environment and to develop safeguards and countermeasures to protect the environment and its users from those risks [3]. Some of the key services security must provide are confidentiality and integrity of interchanged information, authentication of communicating parties, access control and key management.

Confidentiality is the protection of information in the environment from unauthorized access. In smart spaces, the term *information* acquires a unique perspective [4]. The computer systems involved are potentially capable, as a whole, of sensing nearly every aspect of interactions between users and the environment or among users themselves, and all sensed information has the potential of being stored, transmitted, queried and replayed. In smart offices, some of this information will need to be protected due to its business-related sensitivity, but a great deal of the information sensed by the system will be personal information about users. Therefore, apart from the confidentiality issues usually present in information systems, new concerns raise regarding users privacy [5].

Integrity guarantees that only authorized parties are able to modify information in the environment, be it by alteration, repetition, removal or delay of stored information or messages between entities. As with confidentiality, protection of stored and transmitted information has been traditionally achieved through cryptographic means. Code integrity must be provided as well, specially in systems where mobile code is allowed.

Authentication of devices within the smart space can take advantage of existing approaches for computer and network security. Public or private key cryptography can be used to authenticate information interchanges between devices, taking into account the considerations about resource limitations stated above. However, due to the largely decentralized and dynamic nature of smart spaces, key management is the main problem we encounter when dealing with authentication in these environments. Solutions that rely on connectivity to an authentication or revocation server, from Kerberos to public-key certificates, can only be applied to smart spaces where we can assume a hierarchical arrangement of principals, and where addition and removals of principals in the system may be controlled. In [6] such a centralized approach is applied to a smart home, and we will show later its applicability to smart offices as well. In smart spaces where devices communicate through *ad-hoc* networks and where devices need to be added and removed easily *secure transient associations* are used to provide authentication in a distributed manner [7].

Access control intends to ensure that only authorized parties are allowed to perform security-sensitive actions. In smart office scenarios access control policies may become very complex, due to the different roles played by users in the office. One solution to model such scenarios is Role-based Access Control [8], or its extension to take into consideration environmental information defining *environmental roles* [9]. Closely related to access control is delegation [10], which acquire special importance in smart office scenarios.

4 Security Proposal

At the moment no general solution to provide security in smart environments exists, as the security requirements and trust relationships vary broadly in the different kinds of scenarios we may deal with. Therefore, each security solutions must state the assumptions under which it is applicable and the design objectives it pursues. For our agent-based smart office architecture, we assume we can trust the manufacturer of our core smart office architecture, and that connectivity to a centralized building certificate authority (BCA) is available to the SSAP associated to each space in the building, so that a chain of trust from our BCA to the manufacturer of any core service or system agent needed in the building may be established. This trust assumption may not be extended to non-persistent service agents (they are mobile) or personal agents (they enforce user preferences, not system policies). Finally, we assume we can consider the hosts running each SSAP secure. Such security may be granted through the use of Trusted Computing [11] or other secure bootstrap techniques [12].

Also, we require our security solution to support the dynamic nature of the environment (allowing flexible user and device adition and removal), to be scalable, to provide reliable authentication and access control to cope with the rigorous security requirements in smart offices, and to support devices with different capabilities, in terms of power-supply. bandwidth and computing power.

Taking these assumptions and objectives into account, in this section we present our security proposal for the smart office, describing the approach we have taken to address each of the security issues we outlined in section 3.

4.1 Message authentication, confidentiality and integrity

As we have seen in section 3, message security is usually provided using cryptographic means. Asymmetric cryptography is known to provide better security at the cost of more bandwidth and CPU cycles. In our proposal, we assume the use of asymmetric cryptography is acceptable at the SSAPs and at the personal devices (PDAs or smart phones) of the users. However, since personal devices are battery powered, the use of this kind of cryptography should be minimized. Taking this into account our security architecture uses asymmetric cryptography to exchange shared secrets between the communicating parties, using a handshake protocol, which is described in more detail in [2].

4.2 Key Distribution and User Personal Devices

Each SSAP has its own asymmetric key pair, which public key is stored at the BCA. Whenever a new user is added to the system, key pairs are generated for use within the building. For users with access to service personalization, an additional asymmetric key pair is generated for the PA. This allows the system to distinguish between automated requests and direct requests from the user, and also allows the system to ask for user confirmation (e.g. a passphrase) when dealing with sensitive tasks.

4.3 User, Device and Agent Authentication

User authentication is performed by means of certificates. The building must have its own Building-level Certificate Authority Agent (BCAA) to issue Building-level Certificates (BCs) for any user entering the building. A BC associate the user identity to a public key and to a certain number of roles, used to distinguish, for example, an employee from an unknown visitor. These certificates are used to authenticate users to the SSCAs whenever they enter a new space.

Each SSAP has its own asymmetric key pair and its own BC. All system agents and persistent service agents running in the SSAP share this key pair and can use it to authenticate to users, personal agents and other SSAPs and to exchange session keys with them as described in section 4.1. As stated at the beginning of section 4, we assume the SSAP to be secure, and system and persistent agents are code-signed. These agents are thought as the agents of the specific space controlled by the SSAP and so we see coherence in making them share a key associated to that space.

Similar assumptions cannot be made regarding the security of other devices in the smart space, and thus we consider inappropriate to share the SSAP key with these devices and the agents controlling them. Furthermore, some of these devices may not have the computing power, storage space or battery life to put up with asymmetric cryptography. Therefore we use secret key cryptography to communicate with these devices, in a way very similar to Resurrecting Duckling [7]. In our architecture, each SSCA shares a secret key with each device in the space, which allows to create secure transient associations between devices, user and agents within its associated space by assigning temporary shared secret keys to pairs of principals.The same approach is used with non-persistent service agents, since security concerns raised by mobility make sharing the SSAP key pair with them inacceptable. Figure 2 shows this communication mechanism. The personal agent request a video-conference service to the SSCA (1), using a previously agreed session key K_{S1}. After checking the request is legitimate (we will deal with authorization in the following section), a non-persistent service agent NPSA is created (2), *imprinted* to the SSCA by means of a shared secret key, K_{S2}. Then the SSCA generates a temporary session key for the communication between the PA and the newly created service agent, K_{S3}, and sends it securely to both parties (3), which can now communicate using this shared secret key until it expires (4).

4.4 Authorization and delegation

Once users, devices and agents are authenticated and can establish secure communications among themselves, we use a credential-based approach to provide authorization services. The basic idea is that users and agents are allowed to perform an action if they can show a credential signed by a valid authorization authority. In our system this authority is represented by Smart Space Authorization Agents (SSAA). There is one SSAA for each SSAP, and SSAAs may be federated to provide a hierarchical tree of authorization agents.

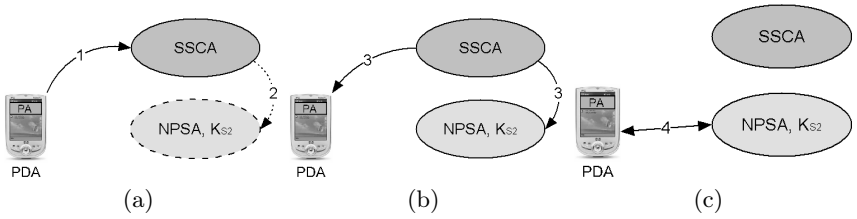

Fig. 2. Secure communication with a non-persistent service agent

There are some additional considerations regarding credential issuing, distribution and storage. Building-level credentials are usually associated to user roles, thus implementing a particular form of RBAC [8]. These role based credentials may be attached to the user building-level certificate, in order to avoid the burden of having to request an specific credential for each request. There may be, however, a limitation for credential storage in the user personal devices. Furthermore, for users not carrying a personal device with enough computing or storage capability, or for those users not having a Personal Agent acting on their behalf, the above protocol is not applicable. For these cases, the system provides an alternative mechanism where the service agents themselves ask the authorization agents for the user credentials. This will be the typical scenario for unknown visitors provided with a smart card to access certain spaces of the building.

Delegation is handled as a particular case of authorization, where the authorisation authority which issues a credential to allow a principal A to perform action X is not an SSAA, but another principal B which is allowed to perform said action. This allows, for example, a personal agent to issue a credential allowing a service agent (e.g. the agent controlling a presentation display) to access a file stored at the user's computer (e.g. to open a slide presentation document).

Both policy definition and certificate and credential management are performed using an agent-based implementation of SPKI [13], as it has been proven as a reliable public-key infrastructure with a good trade-off between expressive power for policy and credential definition and computational load.

4.5 Handling agent mobility

The use of mobile agents raises numerous security considerations [14]. At the moment, our architecture only allows mobility to non-persistent service agents. These agents are created and started by the SSCA upon request of the services they provide, thus allowing certain degree of control of what is running in the SSAP. To increase protection against malicious agents, all non-persistent agents are code-signed by the manufacturer, whose public key is known to all the SSAPs. Whenever a mobile agent tries to migrate to another agent platform, the code signature is verified at the destination to ensure it has not been maliciously

altered. To protect against malicious changes in the state of execution of the agent, the request upon which the agent has been launched is signed by the launching SSAP and attached to the migrating agent, so that the destination platform can restrict what the agent is allowed to do based on the signed request. Furthermore, whenever an agent migrates, its state is signed by the source SSAP, thus taking responsibility of the generated state.

As mobile agents may travel through different SSAPs, even through different buildings, they cannot share the SSAP key as the system and persistent service agents do. Instead, the SSAP generates temporary symmetric shared keys whenever a mobile agent needs to communicate with another principal. Those keys are revoked if the mobile agent leaves the SSAP.

5 Conclusions and future work

There are vastly different lines of research regarding security in smart environments. Though they usually start from the same general assumptions, the strategies they adopt and the importance they associate to each security issue in pervasive computing vary greatly, according to the different scenarios they are intended to deal with. To address the problem of security in a given smart environment, specific requirements and assumptions imposed by the environment must be leveraged to determine the most suitable architecture for the solution.

Smart office environments raise significant challenges regarding security, due to the high number of potential users, devices and spaces, and the diversity of security roles. They usually have more rigorous security requirements. However, smart office environments present some characteristics -detailed security policies, security personnel, policy enforcement among employees- that make possible to achieve a degree of access control unfeasible in other smart environments. We have developed a security solution specifically tailored to the smart office scenario, which takes advantage of the particular characteristics of the environment to satisfy the security requirements.

We believe our solution to be adequately balanced. We have extracted the advantages of federated solutions for security in pervasive computing like Cerberus [15] and distributed solutions like the Resurrected Duckling [7] and put them together to obtain an hierarchical, agent-based solution which is flexible and scalable enough to be applicable to different smart office scenarios, ranging from small businesses to large organizations.

Some issues remain open to further research. We are now implementing new smart office services in our own workplaces to check if the proposed architecture is flexible enough to handle them. Mobile agent security is handled in a very restrictive manner (code signing), and we would like to extend the architecture to support other forms of protection that allow more flexible introduction of new services and devices. Tamper-resistance of the devices as suggested by Stajano should be implemented. Also, we need to address the problem of availability, which has been left apart in this proposal.

References

1. Velasco, J.R., Marsá-Maestre, I., Navarro, A., López, M.A., Vicente, A.J., Hoz, E.d.l., Paricio, A., Machuca, M.: Location-aware services and interfaces in smart homes using multiagent systems. In: Proceedings of the 2005 International Conference on Pervasive Systems and Computing (PSC'05), Las Vegas, USA (2005)
2. Marsá-Maestre, I.: A hierarchical, agent-based architecture for smart spaces. Technical Report TR2006-101, Grupo de Ingeniería de Servicios Telemáticos, Universidad de Alcalá (2006) Available at http://www.it.aut.uah.es/ist/papers/TR2006-101.pdf.
3. Nixon, P.A., Wagealla, W., English, C., Terzis, S.: 11. In: Security, Privacy and Trust Issues in Smart Environments. John Wiley & Sons (2005) 249–270
4. Langeheinrich, M.: Privacy by design: Principles of privacy aware ubiquitous systems. In: UBICOMP 2001, Lecture Notes in Computer Science. Volume 2201. (2001) 273–291
5. Campbell, R., Al-Muhtadi, J., Naldurg, P., Sampemane, G., Mickunas, M.D.: Towards security and privacy for pervasive computing. In: Theories and Systems, Mext-NSF-JSPS International Symposium, ISSS 2002. Lecture Notes in Computer Science, Tokyo, Japan (2002) 1–15
6. Al-Muhtadi, J., Anand, M., Mickunas, M., Campbell, R.: Secure smart homes using jini and uiuc sesame. In: Computer Security Applications, 2000. ACSAC '00. 16th Annual Conference. (2000) 77–85
7. Stajano, F., Anderson, R.: The resurrecting duckling: security issues for ubiquitous computing. IEEE Computer **35**(4) (2002) 22–26
8. Ferraiolo, D., Kuhn, D.: Role based access control. In: 15th National Computer Security Conference. (1992)
9. Covington, M., Fogla, P., Zha, Z., Ahamad, M.: A context-aware security architecture for emerging applications. In: Computer Security Applications Conference, 2002. Proceedings. 18th Annual. (2002) 249–258
10. Na, S., Cheon, S.: Role delegation in role-based access control. In: Proceedings of the 5th ACM Workshop on Role-Based Access Control, Berlín, Germany (2000) 39–44
11. Felten, E.W.: Understanding trusted computing. IEEE Security & Privacy **1**(3) (2003) 60–66
12. W. A. Arbaugh, D.F., Smith, M.: A secure and reliable bootstrap architecture. In: Proceedings of the IEE Symposium on Security and Privacy, IEEE CS Press (1997) 65–71
13. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: Spki certificate theory. RFC 2693 (1999)
14. Jansen, W., Karygiannis, T.: Mobile agent security. NIST Special Publication 800-19, National Institute of Standards and Technology (2000)
15. Al-Muhtadi, J., Ranganathan, A., Campbell, R., Mickunas, M.: Cerberus: a context-aware security scheme for smart spaces. In: Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on. (2003) 489–496