

Security Measures in a Medical Multi-Agent System

Antonio MORENO, David SÁNCHEZ, David ISERN

*Research Group on AI, Multi-Agent Systems Group (GruSMA)
Computer Science and Mathematics Department
School of Electrical and Computer Engineering (ETSE)
University Rovira i Virgili (URV)*

Abstract. This paper describes the main features of an agent-based application that provides medical services to users. The system contains agents that give information about the medical centres, departments and doctors of a city. These units coordinate their execution in order to offer to the user diverse functionalities such as searching for a medical centre, accessing the medical record or booking a visit to be examined by a doctor. Special attention has been paid to the implementation of security mechanisms that guarantee confidentiality in the access and transmission of data.

Introduction

Distributed systems executing in open environments have been extensively used in the last years, mainly because they allow easy access to users, they have a reduced development cost and they can be deployed on the Internet. These systems allow the possibility of accessing multiple services from Internet, from the basic search for information to the execution of commercial or financial transactions. As these systems may deal with confidential data or perform critical actions, it is necessary to apply mechanisms that guarantee privacy and protection in front of possible attacks from malicious users.

The most typical and studied application field in this area is electronic commerce, that has gained great acceptance due to the market possibilities that it offers to industries. However, the access to any kind of service that deals with critical personal data has to be protected with appropriate security measures. In particular, dealing with an individual's health record through Internet is a very sensitive issue. Recently, an American law [17] defined the rules that software companies have to follow in this field, and the penalties that will be applied to those that do not follow these rules. There also exists a Catalan law [6] that defines the access rights of any patient to his/her medical information.

Within this field, we have developed an agent-based distributed application that provides medical services to users through a remote terminal connected to Internet (e.g. a portable PC, a PDA or a mobile phone) [7]. This system tries to ease the access to medical information (data about medical centres, health records) and to offer the possibility of making some transactions remotely (e.g. book a visit to be examined by a doctor). Security issues are clearly very important in this application, and they have been extensively considered. In [8] there is a more exhaustive review of different problems associated to the use of agents in the medical area, including the management of medical records and the

definition of common ontologies. The use of agent technology in the development of this application introduces an innovation in the treatment of security issues. In fact, nowadays there does not exist any universally accepted standard in this area.

This paper provides an accurate study of security problems in a multi-agent system (MAS, [10], [19]) deployed in an open environment (Internet), based on the functionalities provided by the development tool that has been used (JADE [16]). First we explain the features of the implemented system, justifying the choice of agent technology. Then we explain the security model provided by JADE, describing its functionalities but also its important shortcomings. Finally, we explain the security model implemented in our system, detailing the functions provided by JADE as well as the ones that have had to be implemented from scratch to obtain an ad-hoc solution.

1. Description of the Application

The main objectives in the development of the application were the following:

- To design a solution that models accurately the health care organisation, decomposing it in basic units in a three level hierarchy [11] (medical centres, departments and doctors).
- To design an ontology for the medical domain that facilitates information exchange.
- To make the developed components reusable and interoperable with other systems, by using standard protocols and agent communication languages.
- To use security mechanisms that ensure the confidentiality of medical data.

An *intelligent agent* may be defined as a computational process that can perform tasks autonomously. It inhabits a complex and dynamic environment with which it may interact to accomplish a given set of goals [20]. A set of agents that communicate among themselves to solve problems by using cooperation, coordination and negotiation techniques compose a *multi-agent system* (MAS).

Multi-agent systems offer an implementation alternative that certainly fits our needs, because they have the following interesting properties:

- *Modularity*: the different services or functionalities may be distributed among diverse agents, depending on their complexity.
- *Efficiency*: agents may coordinate their activities to perform complex tasks, so that several parts of the same process may be solved concurrently by different agents.
- *Reliability*: any distributed process is more reliable than its centralised counterpart, because there does not exist a single point of failure that may cause the crash of the whole system.
- *Flexibility*: agents may be dynamically created or eliminated according to the needs of the application. Negotiation and knowledge exchange allow the optimisation of shared resources.
- *Existence of a standard*: the FIPA (*Foundation for Intelligent Physical Agents*) [4] is a non-profit foundation based on Geneva (Switzerland). Its main mission is to establish the rules that have to govern the design and implementation of a MAS in order to achieve interoperability among systems. Since 1997 it has been releasing specifications that have been slowly gaining acceptance and have turned into de facto standards in the agents community. Due to this fact, any of our agents is compatible with any other agent that follows the same specifications.

- *Existence of development tools:* JADE (Java Agent Development Environment) [16] is a programming tool that contains a set of JAVA libraries that facilitate the development of FIPA-compliant MASs. Apart from providing low level agent management functionalities and graphical interfaces that ease development and debugging, it also provides an execution environment for agents. Recently, a JADE plug-in that provides certain security mechanisms, called JADE-S, has been released.

We designed and implemented a MAS with the architecture shown in figure 1. This system contains six different types of agents:

- *Personal Agent (PA):* it provides a graphical interface to the user that facilitates the access to the services offered by the system. It is the only agent that can execute outside the main container of the platform and make remote requests through Internet. Figures 2, 3 and 4 show diverse windows of the interface, used to search for information of medical centres, book a medical visit or look at the medical record.
- *Personal Broker (PB):* it provides a gateway among the PA and the other agents of the system. It controls the access to the system and it also checks the identities of the users (see section 3).
- *Medical Centre Agent (MCA):* it models the organisation of a medical centre.
- *Department Agent (DEP):* it models each department (unit dedicated to a medical specialty) within each medical centre.
- *Doctor Agent (DA):* it simulates the behaviour of each of the doctors belonging to the departments of the medical centres.
- *Database Wrapper (DW):* it controls the access to the database that holds the personal and medical information of each patient.

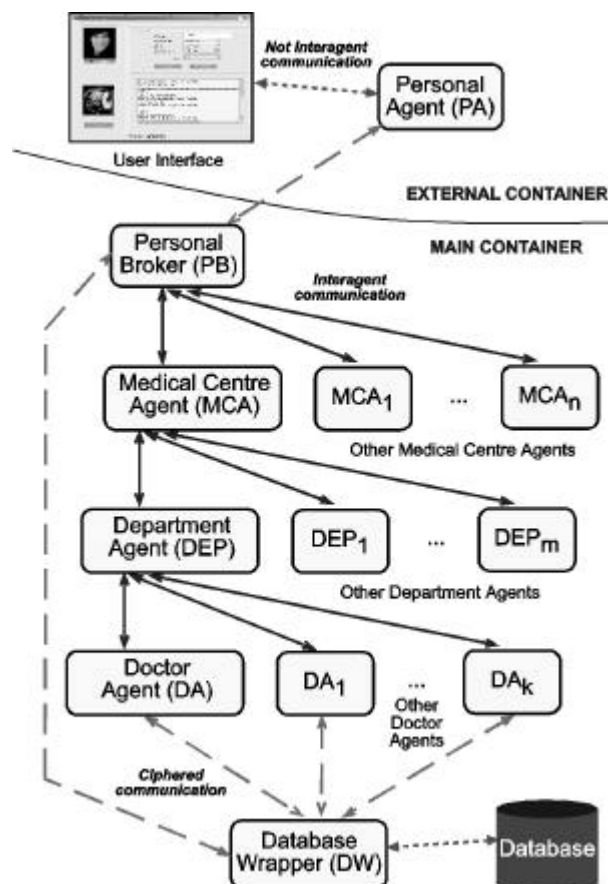


Figure 1. Architecture of the multi-agent system

The basic functionalities provided by the application are the following:

- The user may request information about the available medical centres in a city or region, or about the personnel of a specific medical centre (see figure 2).



Figure 2. Window to request information

- The user can book a visit to be examined by a particular type of doctor in a specific medical centre (see fig. 3).



Figure 3. Window to book a visit to a doctor

- The user has a personal and confidential information area that keeps his/her personal data and health record.

- As mandated by the Catalan Law on Information Access Rights [6], the patient may access the data of his/her medical record (see figure 4).

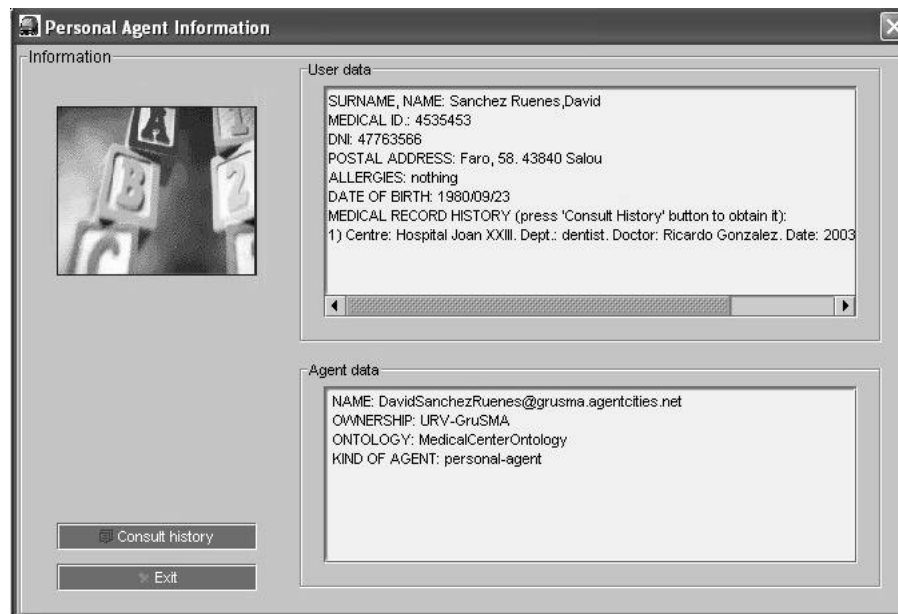


Figure 4. Personal and medical data of the user.

- In the date and time of the scheduled visit, the doctor agent simulates it, storing the results of the examination in the patient's medical record.

2. JADE Security Model (JADE-S)

In order to understand the role of security in the transmission of private and critical information (e.g. health records) through an open environment (e.g. Internet), it is necessary to define the concepts that determine the diverse security levels [3]:

- *Confidentiality*: is the property that ensures that only those that are properly authorised may access the information.
- *Integrity*: is the property that ensures that information cannot be altered. This modification could be an insertion, deletion or replacement of data.
- *Authentication*: is the property that refers to identification. It is the link between the information and its sender.
- *Non-repudiation*: is the property that prevents some of the parts to negate a previous commitment or action.

In the case of a MAS these properties are especially important, due to the autonomy and mobility of agents. A MAS without security support could not be used in an open environment such as Internet if it deals with critical data, because communications could be spied or the identities of the agents could be easily faked.

JADE-S [2] is a plug-in of JADE that allows to add some security characteristics in the development of MAS, so that they can start to be used in real environments. It is based on the Java security model [15] and it provides the advantages of the following technologies:

- JAAS (*Java Authentication and Authorization Service*) [12]: it allows to establish access permissions to perform certain operations on a set of predetermined classes, libraries or objects.

- JCE (*Java Cryptography Extension*) [15]: it implements a set of cryptographic functions that allow the developer to deal with the creation and management of keys and to use encryption algorithms.
- JSSE (*Java Secure Socket Extension*) [14]: it allows to exchange critical information through a network using a secure data transmission (SSL).

2.1 Basic Concepts

A JADE platform may be located in different hosts and have different containers. In order to introduce security in such an open and distributed environment, JADE-S structures the agent platform as a multi-user environment in which all components (agents, containers, etc.) belong to authenticated (through a login and a password) users, who are authorised by the administrator of the system to perform certain privileged critical actions. The general scheme of this environment is shown in figure 5:

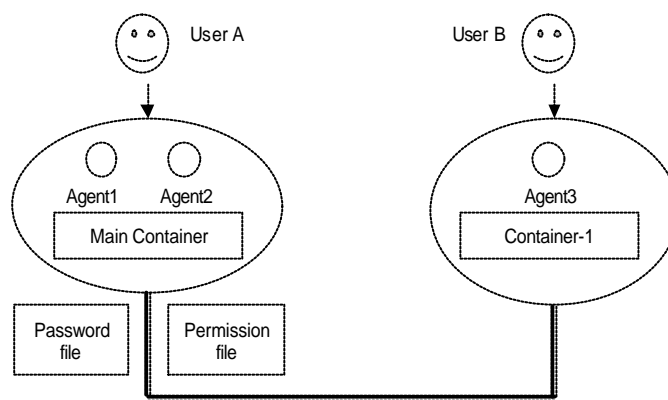


Figure 5. Aspect of a secure agent platform

Thus, in each platform there is a permissions file that contains the set of actions that each user is authorised to perform ([13]).

Internally, an agent proves its identity by showing an Identity Certificate signed by the Certification Authority (provided in a transparent way to the agent when it registers in the system and provides the login and the password of its owner). Using these digitally signed certificates the platform may allow or deny certain actions to each agent.

2.2 Authentication

As explained above, each component of the platform belongs to an authenticated user. The user that boots the platform also owns the AMS¹ and DF² agents and the main container.

When a user wants to join the platform (through one of his/her agents), it has to provide his/her login and password. These data are checked with the passwords file contained in the platform, which is stored in a ciphered way, like Unix passwords. The passwords file is unique and is loaded with the main container.

Each agent owned by this user will have an Identity Certificate that contains its name, its owner and the signature of the Certification Authority.

2.3 Permissions and Access Restrictions

In a JADE-S platform the permissions to access resources are given to the different entities by following the mechanism defined by the new system provided by Java (JAAS [12]) for user-based authentication. Thus, it is possible to assign permissions to parts of the code and

¹ *Agent Management System*: it stores the addresses of the agents and offers a White Pages service.

² *Directory Facilitator*: it is aware of all the services offered by the agents of the system, and it provides a Yellow Pages service.

to its executors, restricting the access to certain methods, classes or libraries depending on who wants to use them. An entity can only perform an action (send a message, move to another container) if the Java security manager allows it. The set of permissions associated to each identity is stored in the access rights file of the platform (which is also unique and is loaded when the platform is booted).

Java provides a set of permissions (apart from those that may be defined by the user) on the basic elements of the language: *AWTPermission*, *FilePermission*, *Socket Permission*, etc. Moreover, JADE-S provides other permissions related to the behaviour of the agents: *AgentPermission*, *ContainerPermission*, etc. Each permission has a list of related actions that may be allowed or denied.

2.4 Certification Authority and Certificates

The Certification Authority is the entity that signs the certificates of all the elements of the platform. To do that, it owns a couple of public/private keys so that, for each certificate, it creates an associated signature by ciphering it with its private key (which is secret). Then, when the identity of an entity has to be checked, the signature may be unencrypted with the public key of the Authority (which is publicly known) and we can check that the identity that the entity wanted to prove matches the one provided by the Authority. The secure platform JADE-S provides a Certification Authority within the main container. Each signed certificate is only valid within the platform in which it has been signed.

2.5 Secure Communication

In order to provide a secure communication between agents located in different hosts or containers, JADE-S uses the SSL protocol (*Secure Socket Layer*) [9] that provides privacy and integrity for all the connections established in the platform. This is a way of being protected against network sniffers.

3. Implementation

In this section we describe in detail all the security mechanisms that have been implemented in the application, including those provided by JADE-S as well as those that have been manually added.

3.1 Access Control

Before using any security mechanism provided by JADE-S on an agent platform, first it is necessary to define a set of default Java permissions that allow to execute the basic code of JADE (including network access, graphical interface, etc.). Recall that the only allowed actions are those explicitly included in the permissions file. These permissions have to appear in the "*basic.policy*" file, which is read by default when the platform is booted.

Having done that, we have built, on top of the security services provided by JADE-S, an access control model that is based on two levels of permissions:

- *Root*: this user has access to all the actions that may be performed on the platform (indicated explicitly one by one in the permissions file).
- *Guest*: the permissions of this user are quite limited. Its agents can only perform the basic actions to request services from the system (send and receive messages).

The content of the passwords file containing these two users has the same format that the UNIX passwords file. This file can be created directly with the UNIX user management capabilities or through the options provided by JADE. The set of permissions of each user

is stored in the access policy file associated to the platform. Thus, we associate one of these users to each of the types of agents:

- **Internal agents:** agents that are executed within the main container of the platform (*Broker, Medical Centres, Departments, Doctors* and *Database Wrapper*). They belong to the *root* user; thus, they do not have any constraint on the actions they can perform. We have taken this decision to facilitate the interaction among them (they can send and receive messages from any agent, or register in the DF or the AMS). As they are internal to the platform (they execute in the main container) and they have been programmed by us, they will not perform any malicious action (kill other agents, deregister other agents, fake the identity of another agent, etc.).
- **External agents:** the *Personal agents*, that execute in an external container in another host, belong to this category. They have been associated to the *guest* user, so that they can neither access the main container (to join the other agents) nor access the DF (to modify the information of other agents). They can only communicate with the internal agents through the *Broker*; therefore, they cannot pretend to have the identity of other agents or kill other agents. All these constraints are necessary because we do not have any control over these agents, and they may have been programmed to perform dangerous actions.

3.2 Secure Communication

Apart from the security mechanism implemented through the permission file, we have also taken advantage of the possibility offered by JADE-S of ciphering all the communications using SSL [9].

Even though it is only necessary to encrypt those messages between the main container and the external container (between the *Personal Agent* and the *Broker*) that contain confidential information (medical records), the activation of SSL may only be made globally: we can only cipher all messages or none of them.

Thus, by activating the appropriate option in the JADE initial configuration files in the client (*Personal*) and in the server (platform), *all the communications between the agents of the system will be ciphered* and, therefore, we can safely send the encryption keys or the medical records. This mechanism works if we have previously obtained an identity certificate for the server side (the one that boots the platform), so that SSL may implement authentication. Having this certificate, we can use the *keytool* application (included in the Java distribution) to create a couple of files called *keystore* and *truststore* that have to be included in the execution directory. Certificates are provided freely (for evaluation) by some certificate companies (e.g. Verisign [18]).

It must be remembered that SSL is a protocol that works at the data transport level and, therefore, is transparent to the application. Thus, even though in the development phase we do not have to take it into account, it does not allow us to access the security mechanisms that it incorporates (e.g. the possibility of authentication using certificates).

3.3 JADE-S Shortcomings

Despite its functionalities, it must be stressed that JADE-S is still in an early phase of development and it presents some constraints such as the following:

- SSL can only be globally applied to all messages (not individually or selectively).
- The access permissions are quite generic and limited, and do not allow to define specific actions (e.g. to allow an agent to make queries to the DF but not register in the DF).

- It is not possible to access the information about the identity of the agent from the program.
- We can only define the permissions specified by JADE-S. It is not possible to add new permissions related to the user-defined agents (e.g. about the possibility of requesting a service).

Due to these shortcomings, we have had to add some software security mechanisms to create a reliable and flexible security model.

3.4 Centralised Access Model

One of the main problems of JADE-S is that the permissions that can be defined on the DF (agent that provides information about all the other agents that are executing on the platform) are reduced to the possibility of allowing the reception/sending of messages. However, several actions may be performed on the DF: an agent may register/deregister, or search for other agents that satisfy a given constraint (e.g. that offer a specific service or belong to a certain class). Taking into account that the registration/deregistration operations are critical and that there does not exist any control on the identity of the agent that requests the action, we have had to forbid the access to external agents (*Personal Agents*) to the DF to prevent them from making these actions maliciously (pretending to be other agents). The problem of this approach is that we will no longer be able to search for agents directly, because the communication with the DF is not allowed.

The solution has been to implement a new behaviour in the *Personal Broker* that simulates the functionality of the search process as if it was made directly by the DF. In this way the *Personal Agent* has the possibility of searching for agents without having to talk directly to the DF (the *Broker* provides the gateway).

Another shortcoming of the JADE-S security model is that it does not allow a user to define his/her own permissions for each type of agent (e.g. to control the access to the service of booking visits with doctors). Thus, if we allow the *Personal Agent* to access the *Medical Centres* and the *Database Agent* directly to make a booking or to request information, we can only indicate if the PA can send/receive messages from these agents, but not specify the services that may or may not be requested. Furthermore, in this decentralised model it is more difficult to guarantee security, as it should have to be implemented at different levels. Therefore, we have decided that *Broker* provides a gateway between the *Personal* and the rest of the system, forbidding the communication with any other agent. In this way we can prevent the *Personal* agent from faking the identity of a *Department* when it communicates with a *Medical Centre*, and agents do not have to control the identity of all the senders of the messages they receive.

This centralised model allows us to incorporate all the program security mechanisms described below in a single agent. In this way the rest of the system is not involved in this control and we get a transparent use of the system by external agents, as they only have to communicate with the *Broker* and not with particular agents representing medical centres, departments or doctors. The price to be paid is that the *Broker* has to implement indirectly all the services that are offered to the user, and its implementation gets more complex.

3.5 Software Authentication

JADE-S deals internally with the identities of the users, but this information is not accessible from the program and, as we need to know this information (e.g. to provide the medical records of the users), we had to implement an authentication mechanism.

Classical authentication is based on public key algorithms (in our case RSA), so that each user owns a public key and a private one (which is only known by the agent itself and

by the certification authority that generated it). In our case, the keys (managed by the *Database Agent*) are associated to the user from the personal data that it provides when it enters the system. These keys allow the agent to sign the messages that it sends, so that its identity may be checked.

When the user joins the system, the *Database Agent* generates its keys and sends them through the *Broker*. This agent stores the public key of each registered user and sends both keys to the *Personal Agent*. It is not necessary to control in the program the identity of the agent that sends the message (*Database* or *Broker*) with identity certificates, because it is controlled at a lower level (using SSL and JADE-S users management). When the user wants to send a critical message in which it has to prove its identity (e.g. when it wants to access the medical record or request a visit to a doctor), it encrypts the message with its (secret) private key. When *Broker* receives this message, it can check the identity of the sender by using the public key associated to this agent. If the unencrypted content is valid, the identity is deemed correct and the request is processed. If the key does not match the one of the agent or the content of the message has been modified, the result of the unencryption will be wrong and will provoke an exception during the interpretation process, which will cause a denial of the request (see figure 6).

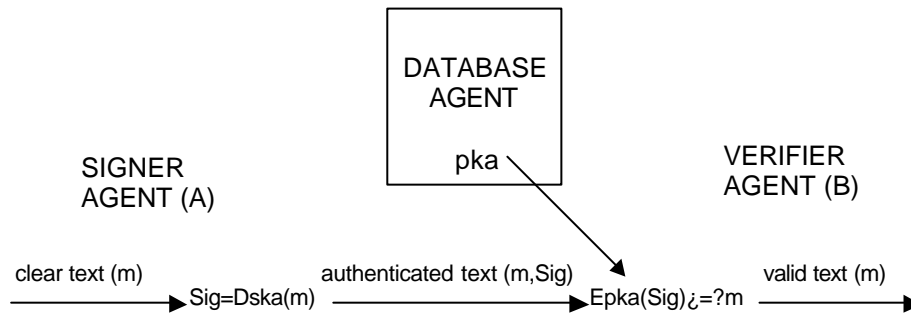


Figure 6. Signing a message with a public key

This mechanism is also used by *Doctors*, to check that only they are allowed to modify the medical records of the patients when they perform a medical examination. In this case, though, keys are not stored, because their data are read statically from a configuration file and are not stored in the database. Thus, keys are valid during the life cycle of the *Doctor* agent (they work as “session keys”).

4. Summary and Conclusion

The main features of our security model are the following:

- All the messages exchanged among the agents are encrypted at the transport level, and the server is authenticated with SSL.
- We have established diverse security levels by using different user identities, and associating an owner (identified with a login and a password) to each agent.
- We have defined a set of permissions for each security level through descriptor files.
- It is a centralised access model through the *Broker* agent, that implements a software security control.
- Personal agents cannot communicate directly with any other agent (even with the DF).
- User authentication through the signature of critical messages using a public key mechanism.

We want to argue in this paper that MAS are a good alternative to fully implement complex programs in distributed environments such as Internet. In fact, a complete prototype of the application is permanently running on a server provided by AgentCities in a worldwide agent-based network of services. The project web page [6] contains all the access instructions to be followed to request securely and remotely the services offered by the system.

Even though there is still a lot of work to be done in the security field in MAS, this paper tries to show that it is feasible to apply concepts of information security in these systems. Thus, if FIPA continues developing standards in this field [5] and development tools incorporate these suggestions, it will become possible to develop secure applications in critical domains such as banking or on-line electronic commerce.

Acknowledgments

This system has been developed with the support of *AgentCities.NET* through the deployment grant "Deployment of agent-based health care services" ([1]). The authors also acknowledge the support of the Spanish thematic network "Creación de un entorno innovador para la comunicación de agentes inteligentes" (MCyT, TIC2001-5108-E).

References

- [1] GruSMA web page for AgentCities project: <http://grusma.etse.urv.es/~agentcities>.
- [2] Bellifemine, F. et. al. *JADE Security Administrator Guide*, CSELT S.p.A. and University of Parma, September 2002 (v 2.61), <http://sharon.cselt.it/projects/jade/>
- [3] Domingo, J., Herrera, J., *Criptografia per als serveis telemàtics i el comerç electrònic*, EdiUOC, 1999.
- [4] Foundation for Intelligent Physical Agents, web page: <http://www.fipa.org>
- [5] Foundation for Intelligent Physical Agents (2002) *FIPA MAS Security white paper*, FIPA
- [6] Generalitat de Catalunya. *Llei sobre els drets d'informació concernent la salut i l'autonomia del pacient, i la documentació clínica*. Law 21/2000, 29th December 2000, Departament de Salut i Seguretat Social, Generalitat de Catalunya. Quaderns de legislació, 31. ISBN 84-393-5459-2.
- [7] Moreno, A., Isern, D., Sánchez, D., *Provision of agent-based health care services*, AI-Communications, in press, 2003.
- [8] Nealon, J. Moreno, A. *Agent-based health care systems*. In *Applications of Software Agents Technology in the Health Care Domain*, J.Nealon and A.Moreno Eds., Whitestein series in software agent technology, 2003 (in press).
- [9] Netscape Communication Corporation, *Introduction to SSL*, available at the web page <http://developer.netscape.com/docs/manuals/security/sslin/contents.htm>
- [10] Poggi, A., Rimassa, G., Tomaiuolo, M., *Multi-User and Security Support for Multi-Agent Systems*, DII – Università de Parma, Proceedings of WOA 2001, Modena, Sep 2001.
- [11] Servei Català Salut: <http://www.gentcat.es/scs>
- [12] Sun Microsystems, *Authentication and Authorization Service (JAAS)*. Web page: <http://java.sun.com/products/jass/index-14.html>
- [13] Sun Microsystems, *Java Default Policy Implementation and Policy File Syntax*, <http://java.sun.com/j2se/1.4/docs/guide/security/PolicyFiles.html>
- [14] Sun Microsystems, *Java Secure Socket Extension (JSSE) Reference Guide*. Web page: <http://java.sun.com/j2se/1.4/docs/guide7security/jsse/SEERefGuide.html>
- [15] Sun Microsystems, *Java Security*. Web page: <http://java.sun.com/security>
- [16] TILab, Jade, <http://sharon.cslet.it/project/jade>

- [17] US Department of Health and Human Services, *Standards for Privacy and Individually Identifiable Health Information*, Federal Register, volume 65. December 28th, 2000.
- [18] Verisign, plana web: <http://www.verisign.com/>
- [19] Wong, H., Sycara, K., *Adding security and trust to multi-agent systems*, In Proceedings of Autonomous Agents'99, Workshop on deception, fraud and trust in agent societies, pp. 149-161. Seattle, Washington, May 1999.
- [20] Wooldridge, M., *An introduction to multiagent systems*, John Wiley Ed., 2002. ISBN 0-471-49691-X.