Secure Mobile Agent System and its Application in the Trust Building Process of Virtual Enterprises

Michelle S. Wangham[1], Joni S. Fraga[1], Ricardo J. Rabelo[1] and Lau C. Lung [2]

[1] Department of Automation and Systems - Federal University of Santa Catarina

C.P. Box 476 - CEP. 88040-900 - Florianópolis – SC – Brazil

Phone: +55(48) 331.7675 - Fax: +55(48) 331.9770

{wangham, fraga, rabelo}@das.ufsc.br

[2] Graduate Program in Applied Computer Science

Exact Sciences and Technology Center

Pontifical Catholic University of Paraná - Curitiba - PR - Brazil

Phone: +55(41)3271.1347 - Fax: +55(41)3271.1669

lau@ppgia.pucpr.br

Abstract

The paradigm of mobile agents may be used in large-scale distributed systems due to its flexibility that comes from the mobility notion. However, this promising technology brings some security concerns of its own. This paper focuses on mechanisms of the Mobile Agent Security Scheme – *MASS* – for protecting mobile agents against malicious platforms by considering large-scale systems. This proposed scheme, based on secure data repositories, comprises prevention and detection cryptographic protocols of agent security violation. The scheme is based on the SPKI/SDSI infrastructure and on the concept of SPKI Federation for providing scalability and flexibility to the distributed applications on the Internet. In addition, it proposes an approach on how trust building in mobile agent-based architectures can be reinforced by the use of security mechanisms in the search and select of partners for creating a Virtual Enterprise. A prototype of the *MASS* was implemented and integrated in order to demonstrate its suitability for a distributed application with mobile agents (*MobiCII* system).

1. Introduction

A mobile agent in a large-scale network can be defined as a software agent that is able to autonomously migrate from one host to another in a heterogeneous network in order to perform tasks on behalf of its application [14]. For these agents exist within a system or to compose a system itself, they require a computing environment - *an agent platform* - for deployment and execution. An agent is created in its *home platform*, which is usually considered a trusted environment for the agent. The mobile agent can either follow a pre-assigned itinerary on the network or determine its itinerary dynamically based on information gathered from the network. An agent has its own thread of execution, and thus acts independently of its home platform.

Mobile agents can be used in distributed systems for information searching, filtering and retrieval, for system administrator tasks, for electronic commerce on the web, among other tasks [3]. In spite of its advantages, the mobile agent paradigm introduces new security threats [24]. Due to these threats, security mechanisms should be designed to protect the communication infrastructure, agent platforms and agents themselves. Mechanisms currently available for reducing the risks of this technology do not efficiently cover all the existing threats. Moreover, they introduce performance restrictions that frequently outweigh the benefits from the use of mobile agents for certain applications. To address these concerns, a *Mobile Agent Security Scheme* (***MASS*)** was developed, based on cryptographic protocols and on decentralized authentication and authorization controls that use SPKI/SDSI certificates [4]. In light of the above mentioned, this paper focuses on the *MASS'* mechanisms for protecting mobile agents against malicious platforms by considering large-scale distributed systems. The *MASS* includes prevention and detection techniques to protect mobile agent systems.

A large variety of Collaborative Networks (CNs) has emerged during the last years as a result of the challenges faced by both the business and scientific worlds, such as Virtual Enterprises (VE), Professional Virtual Community (PVC) and Virtual Laboratory (VL) [12]. Within the CN scenario, cooperation in the form of Virtual Enterprises (VE) represents a modern strategy which has been adopted by many enterprises, professionals and laboratories around the world in order to accomplish a given business opportunity, to take part in new markets and to reach scientific excellence for innovative developments. Actually, a VE corresponds to a dynamic, temporal and logical aggregation of autonomous entities (companies, people, governmental institutions, etc.) that interact with each other as a strategic answer to attend, for instance, a given opportunity. The selection of the most suitable VE members has

been often supported by partner's search and selection systems that are applied over a pre-defined group of organizations - a VE Breeding environment (which can be seen as an evolution of the cluster concept) [20].

However, despite this trend for collaborative works, most organizations (companies and/or professionals) are still skeptic to share sensitive information with previously unknown partners. Actually, collaborative network demands the development of relationships with a broad range of potential partners, each one having a particular competency that complements the others. Therefore, it is essential for the effectiveness of the VE creation process that *trust* among the partners be built. Taking these aspects into consideration, this article addresses how the *MASS'* mechanisms can be efficiently applied to a mobile agent-based application for searching and selecting partners in the creation of Virtual Enterprises (VEs) – the *MobiC-II system* [19].

Aiming at greater efficiency and flexibility, this paper brings an approach, which is based on an agent-based hybrid architecture (mobile and stationary agents), to aid the creation phase of VEs. This work comprises security aspects for the use of mobile agents to strengthen the trust building process during the creation of a VE. The security threats within this scenario are also analyzed and a security policy that aims at mitigating most of these threats is defined. In the proposed scheme, the flexibility needed for the implementation of this security policy is given by the ability to select only the subset of the *MASS'* mechanisms desired by this application.

The remaining contents of this article are organized as follows. Section 2 discusses the security problem in mobile agent systems and some approaches to mitigate these threats. Additionally, it presents an overview of the MASS' mechanisms and its security objectives. Section 3 introduces the partner search and selection system (*MobiC-II*), discusses the security threats to a real scenario and explains how these mechanisms are applied to *MobiC-II*. Section 4 describes the prototype results and discusses its performance. Section 5 reviews the literature related works in this subject. Finally, Section 6 contains the conclusions and suggestions for further research.

2. Security in Mobile Agent Systems

One of the main concerns about the implementation of an agent platform is to ensure that agents are unable to interfere with an underlying agent platform. Mobile agent platforms face several threats, such as [26] masquerading, denial of service, unauthorized access and repudiation. The establishment of isolated protection domains for each incoming mobile agent, and the control of all inter-domain access is an

approach that has been commonly adopted, so as to offer protection to agent platforms. In addition to these approaches, other techniques are also used for improving the platform security against malicious agents. Some of these techniques are safe code interpretation [21], digital signatures [21], path histories [5], State Appraisal [25], and Proof-Carrying Code (PCC) [9].

Attacks performed by malicious platforms on agents are the most difficult security problems to overcome and have not been provided with an appropriate solution yet. While mechanisms for the platform security are a direct evolution from traditional mechanisms that emphasize prevention techniques, mechanisms directed towards agent security usually correspond to detection measures. This occurs due to the fact that an agent is totally susceptible to a platform and that it is difficult to prevent the occurrence of malicious behaviors. The security of mobile agents mainly involves (*i*) the agent integrity, in order to detect or to prevent platforms from altering the code or data of agents, and (*ii*) the confidentiality of the code and of the agent state, in order to avoid the violation of the intellectual property. With digital signatures, it is possible to protect only the code of the agent and the integrity of origin data (read-only data generated during the agent creation). However, in the case of multi-hop mobile agents, it is very hard to ensure the integrity of data generated in visited platforms. Due to this fact, there are few commercial and academic platforms that deal with the security issue of mobile agents.

The prevention approaches based on Secure Hardware guarantee the integrity of the agent. However, these approaches are not very practical because of the high costs involved. Equally, the Computing with Encrypted Functions approach [22], which aims at ensuring the confidentiality of the agent during its computation, has not showed practical solutions yet. Also, the Code Obfuscating approach [7], due to the difficulty in measuring its efficiency and protection time, usually has its applicability limited to applications that require soft security.

The detection techniques seem to be the most suitable ones to be applied to mobile agent security. Nevertheless, some of the limitations found in techniques such as Itinerary Records with Replication and Voting [6], and Cryptographic Traces [10] showed that these techniques still need to be carefully examined before being considered totally applicable. In [19], two techniques are proposed to protect the mobile agent using an embedded mark in the mobile agent – Watermarking and Fingerprinting. However, as presented by the authors, some attacks are not detected and several drawbacks can be identified in these approaches. Partial Result Encapsulation [8, 16], Read-Only Container and Targeted State approaches [16] are the techniques that have the best results [23].

The Partial Result Encapsulation approach aims at encapsulating the results of an agent's actions, at each visited platform, for subsequent verification. In general, there are three alternatives to encapsulating partial results [26]: (1) providing the agents with a means for encapsulating the information; (2) relying on a trusted third party for this encapsulation (TTP); or (3) relying on the encapsulation capabilities of the agent platform. The first one may have many limitations that are related to the manipulation of keys and key generation functions by the agent [8]. The second alternative, besides causing loss in performance – due to the agent's need of migrating to a TTP every time that it visits an unreliable site – has other limitations and vulnerabilities as described in [23]. The third option - which depends on the platform to run the encapsulation – has proved to be the most suitable one. The following studies pursue this approach: KAG Family of protocols [8], Append-Only Container [3, 16], and Multi-Hops Protocol [2]. In Section 6, these studies are compared to the *MASS'* proposal.

2.1 An Overview of the MASS: A Mobile Agent Security Scheme

The security scheme - *MASS* - is based on a model of agents that assumes free itineraries and multi-hops. The Mobile Agent Facility (MAF) specification [17] is used as a guideline to achieve interoperability between mobile agent systems. The *MASS* is composed of security mechanisms designed to protect the communications infrastructure, agent platforms and agents themselves. The scheme is based on the SPKI public-key infrastructure [4] and the concept of SPKI federation [1] for providing scalability and flexibility to distributed applications. Figure 1 shows the prevention and detection procedures defined in the *MASS*.

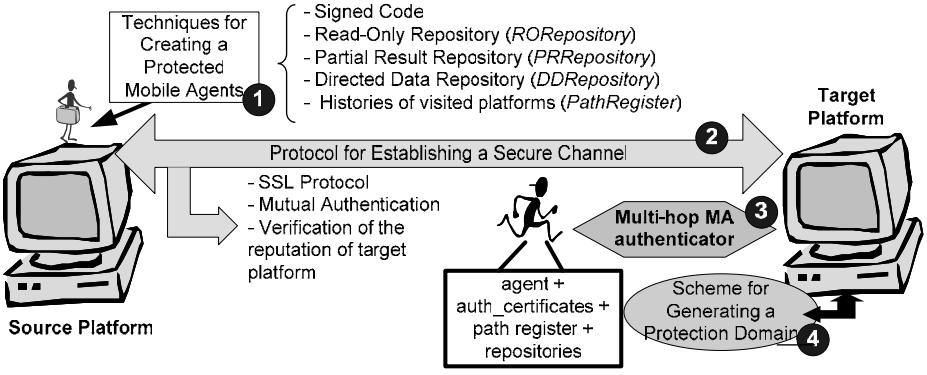

Figure 1- *MASS*: Security Scheme to Mobile Agent Systems

2.1.1 Techniques for Creating a Protected Mobile Agent (Step1, Fig. 1)

During the creation process of a mobile agent, the owner – the authority that an agent represents – provides a set of SPKI/SDSI authorization certificates defining the privilege attributes of the agent. The owner of the agent has to generate an object (called *Path Register*) that contains a signature indicating agent identity and the identity of the first platform to be visited. This object is attached to the agent. Also, agents can have a platform list attached that indicates which platforms are authorized to execute the agent. The agent programmer can protect items in the state of the agent using secure data repositories. All these operations are executed before creating the agent in its home platform. The data repositories are discussed in the next section.

2.1.2 Protocol for Establishing Secure Channels (Step 2, Fig. 1)

In the *MASS*, before agents can be migrated, mutual authentication between the sending and receiving platforms must be established, which creates a secure channel in the communication infrastructure. The mutual authentication is performed via a Challenge/Response protocol based on SPKI/SDSI certificates [4]. The establishment of a secure channel remains valid in the subsequent interactions between the involved platforms. An underlying security technology (Secure Sockets Layer - SSL) is used to ensure confidentiality and integrity of the communication between agent platforms through secure channels.

2.1.3 Mobile Agents Authentication (Step 3, Fig.1)

Before instantiating a thread to an agent, the target platform must authenticate the received agent. A *multi-hop authenticator* was defined in order to establish trust on an agent, which is based on the authenticity of the agent owner and of the platforms visited by the agent. As a platform receives a mobile agent, it must first check that this agent has not been corrupted and then confirm its association to a principal, that is, to its owner. The multi-hop authenticator is described in Section 2.2.7.

2.1.4 Procedure for Generating Protection Domains (Step 4, Fig. 1)

Protection domains and the permissions assigned to them are defined after the trust on an agent has been established in the receiving platform. Protection domains are generated based on the SPKI/SDSI certificates. The authorization certificates carried by an agent need to be verified by the platform guardian in order to generate permissions and the corresponding protection domains to the agent. This scheme

decouples the privilege attributes granted to principals (mobile agents) from the attributes required to access resources protected by the platform (control attributes or policies), and thus offer a more flexible and dynamic access control for large-scale systems with respect to the Java access control.

2.2 MASS$_{ma}$: Protecting Mobile Agents with Secure Data Repositories

According to the target of security, the MASS was divided into two sub-schemes:

- *MASSap* - a scheme to protect the agent platforms and the communication infrastructure, and
- *MASSma* - a scheme to protect mobile agents.

The description of the *MASS*$_{ap}$ can be found in [14, 15]. Figure 2 illustrates the procedures, defined in the proposed security scheme, which emphasize the protection of mobile agents in large-scale systems. The boxes in Figure 2 describe techniques that can be selected by a programmer for agent protection. Prior to the sending of a mobile agent, the source platform can verify, with the aid of public information, the reputation of the target platform (box 1, Fig. 2). This reputation is defined by a social control mechanism [13], which is based on the infrastructure of the SPKI Federations. Despite the fact that the authentication of mobile agents is a technique that aims to protect platforms against malicious agents, it can be also used (*i*) to detect non-authorized modifications in an agent during its hops and (*ii*) to aid the identification of the visited malicious platforms (box 2). When the aim is to protect the agent state, the programmer can select among the proposed techniques for such a goal (box 3). During the agent authentication, when a violation of the agent integrity is detected, the mechanisms of social control have to be activated for identifying malicious platforms and for updating platform reputations (oval 4).

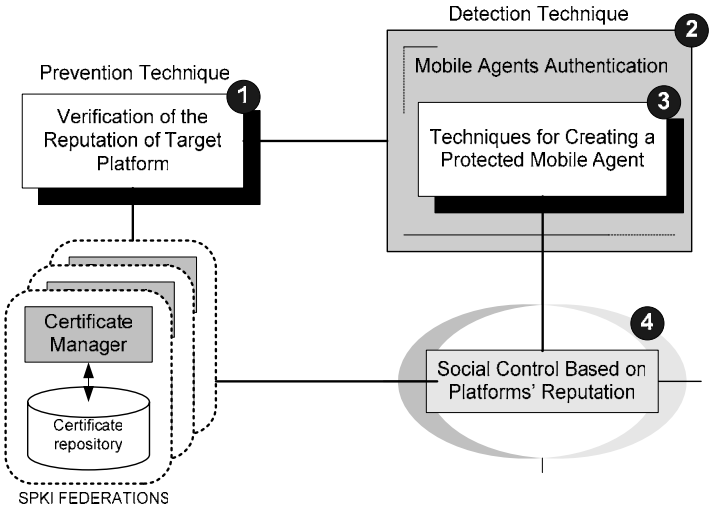

Figure 2 - *MASSma*: Protecting Mobile Agent in the *MASS*

In the proposed scheme, mobile agent platforms can be grouped according to their service classes, constituting service federations and defining trust relationships among themselves. The **concept of federations**, introduced in [1], aims to facilitate the access of clients to services through a dynamic establishment of trust chains. These trust chains between client and service are quickly and efficiently established from name and authorization SDSI/SPKI certificates available in the certificate repository of the service federation. Besides, different federations can establish trust relationships by creating federations Web with a global scope. The main function of a federation web is to help a client, through its agents, search for access privileges that link it to the guardian of a service (another platform). Further details on the concept of federations can be found in [1].

In the proposed scheme SPKI federations are used not only to find certificates and help build new certificate chains among platforms, but also to assist the agent to establish the trust on a platform. When an agent is created, a list indicating the service federations, whose member platforms are authorized to execute the agent, can be defined and attached to the agent. In addition, managers of the federations are responsible for keeping a list containing the reputation of the platforms, which belong to a given federation. Based on this information, platforms can be excluded from the federation due to misbehavior.

In the $MASS_{ma}$, a platform has to define the quality of protection required to the agent while it creates a mobile agent, before dispatching it to the first target platform. This **quality of protection** (*QoP*) is a unique read-only attribute that expresses the security mechanisms that have to be used by the platforms visited by the agent.

2.2.1- Structure of a Mobile Agent

Besides **Code Signing** used for protecting the agent code, the *MASS* provides three repositories with the purpose for detecting possible violations of the mobile agent integrity. The *Read-Only Repository* is used to maintain immutable data. The *Partial Results Repository* is where partial results collected during the hops of the agent are stored. The third technique in the $MASS_{ma}$ is the *Data Directed Repository*, which aims not only for data integrity but also for confidentiality of some data so that they are not revealed to non-authorized platforms. The proposed data repositories are distinguished in the agent state, as Figure 3 shows.

Many agent platforms use only the signature of the mobile agent code to assert agent ownership. However, Roth [23] states that digital signatures attached to an agent code are not sufficient to distinguish

one agent instance from another. The owner should sign static data, which include the code of the agent as well as enough cryptographic redundancy to distinguish between two instances of the same agent. In the *MASS*, once the agent program is created, the programmer needs to generate the static data to the agent, called agent's credentials. A cryptographic hash of the agent's credentials serves as a unique identifier that should be aggregated to the agent repositories to provide the uniqueness of the agent instance.

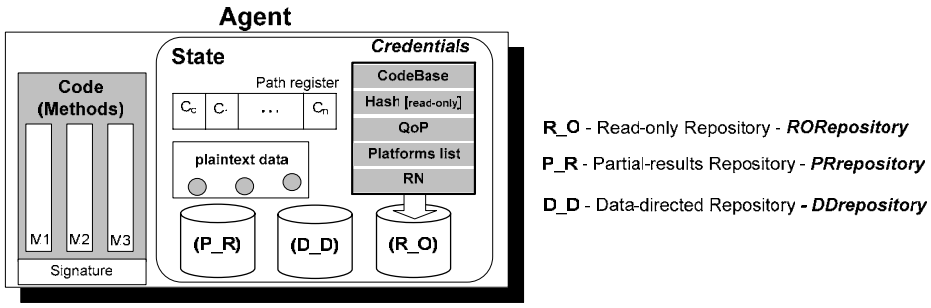

Figure 3- Proposed Structure to a Mobile Agent

Each agent carries its signed Credentials as part of its state, inside the Read-Only Data Repository (*RORepository*). The Credentials object is composed of five parts (see Fig.3):

- **Codebase**. It specifies the location (URI) of the server of the classes that an agent may require.
- **Read-only data hash**. The main objective of this *hash* is to link the Credentials with the initial data of the agent, stored in *RORepository*. The value of this cryptographic hash of the read-only data is also stored in the Credentials object. This link is useful to detect attacks in which a malicious platform reuses the credentials of a visiting agent in another agent.
- **Quality of protection (QoP)**. It is an attribute that identifies the quality of protection required that must be satisfied by all the platforms visited by an agent.
- **SPKI Federations list**. It is an attribute which indicates the SPKI federations, whose member platforms are authorized to run the agent.
- **Random number** (*RN*). It is a *nonce*.

In the proposed structure for a mobile agent shown in Figure 3, a path register object and the three data repositories can be part of the agent state. The repositories are used to safely store the data collected or carried by the agent.

2.2.2- Verifying the Platform Reputation

In order to prevent the sending of a mobile agent to a platform that is not a member of one of the authorized federations, the sending platform also verifies the target platform reputation. The sending platforms are responsible for meeting the quality of required protection and defining the level of trust of the agent in the next platform to be visited. The proposed mechanism for verifying reputation depends on social control policies which attempt to locate suspect platforms and update the reputation of each platform present in the list stored in the federations. The federations must provide mechanisms for the automatic query of these reputation lists.

So as to check the platform, the mechanism (*i*) requests the *black* lists of the federations and (*ii*) analyzes these lists and defines the trust on the target platform. A platform is considered untrustworthy when it is found in one of the black lists aforementioned[1]. Step *i* may add significant costs to the performance of some applications. For that matter, it has been defined that the lists sent by the federations can be stored for a certain period of time in a platform, which eliminates the communication costs for each sending of agent in the platform. These lists have to be periodically updated.

2.2.3- Read-Only repository

In the proposed scheme, an owner must sign the code of its agent in order to protect its integrity. In addition to the code, all the data indicated by the agent programmer as being read-only can be also signed and stored in the RORepository. The technique employed in this study is based on the Read-Only Container proposed in [16]. Besides the specific data of each application, the read-only repository will contain the Credentials object. Any platform visited by the agent can verify the integrity of this repository.

2.2.4- Partial-Results Repository

In the $MASS_{ma}$, a set of cryptographic protocols is used by the platforms to insert and to protect the sensitive data that each platform has generated. These sensitive data are stored in the *PRRepository*. The proposed protocols are extensions of P1 and P2 KAG protocols, defined by Karjoth et al. [8]. Some adaptations and modifications have been made to P1 and P2 protocols. In accordance with the notation

[1] If a target platform is in a federation's black list, it is considered *untrustworthy* and the agent is sent back to its home platform.

presented in Table 1, Figure 4 describes the necessary procedures for the *PRRepository* initiation and for the insertion of the partial result calculated in each platform.

P_o	Home Platform
P_i , $1 \leq i \leq n$	Platforms visited by a mobile agent
r_i	A nonce generated by P_i
$ENC_o(key_i)$	Temporary key encrypted with the public key of P_o
$ENC_{keyi}(m)$	Message *m* encrypted with the temporary secret key key_i created in P_o
$SIG_i(m)$	Signature of P_i on a message *m*
$H(m)$	A one-way hash function (e.g., SHA-1)
$P_1 \rightarrow P_2$: *m*	P_1 sending message *m* to P_2
pr_i	Partial result obtained in P_i
PR_i	Protected partial result encapsulated in P_i
h_i	Value of chaining relation in P_i (to PR_i)
$H(creds)$	Hash value of Credentials (agent identifier)

Table 1- *MASS'* Cryptographic Notation

To **initiate the *PRRepository***, the owner of the agent (in platform P_o) picks a random number r_o. Next, it computes the value h_o by applying a hash function over r_o and over the identity of the first platform to be visited (P_1) (step 1 in Fig.4.a). This h_o is the initial value of the chaining relation among the partial results. After that, the value r_o is encrypted using the public key of the agent owner. The protected partial result PR_o is generated by taking the encryption result, the unique identifier of the agent ($H(creds)$), and the chaining relation h_o and signing all these data with the owner private key (step 2, Fig. 4.a). Finally, in step 3, PR_o is inserted in the *PRRepository* of the agent and (it is) sent to P_1.

In order to insert partial results in the *PRRepository,* the owner of the agent must define which protocol will be used by the agent platforms according to the needs of each application. It must also inform about the protocol used by the platforms to be visited. This information should be stored in the agent's credentials, in the attribute which identifies the quality of protection (*QoP*).

Protocol A (see Figure 4.b), used to insert partial results, is based on the protocol P1 from the KAG Family. The first step computes the hash value over the previous partial result, concatenated with the identity of the next platform. The chaining relation (h_i) has two purposes. First, it links the previous partial results with the current partial result. Thus, PR_{i-1} cannot be modified unless PR_i is modified as well. Second, the inclusion of the identity of the next platform guarantees that only P_{i+1} is able to append the next partial result. In step 2 (Fig. 4.b), in order to keep the confidentiality of some sensitive data

generated in the platforms, each platform must encrypt, through a temporary secret key[2], the partial result together with the random number generated by the platform itself. In order to construct the protected partial result (PR_i), the platform should sign the encryption results concatenated with h_i and the *H(creds)*.

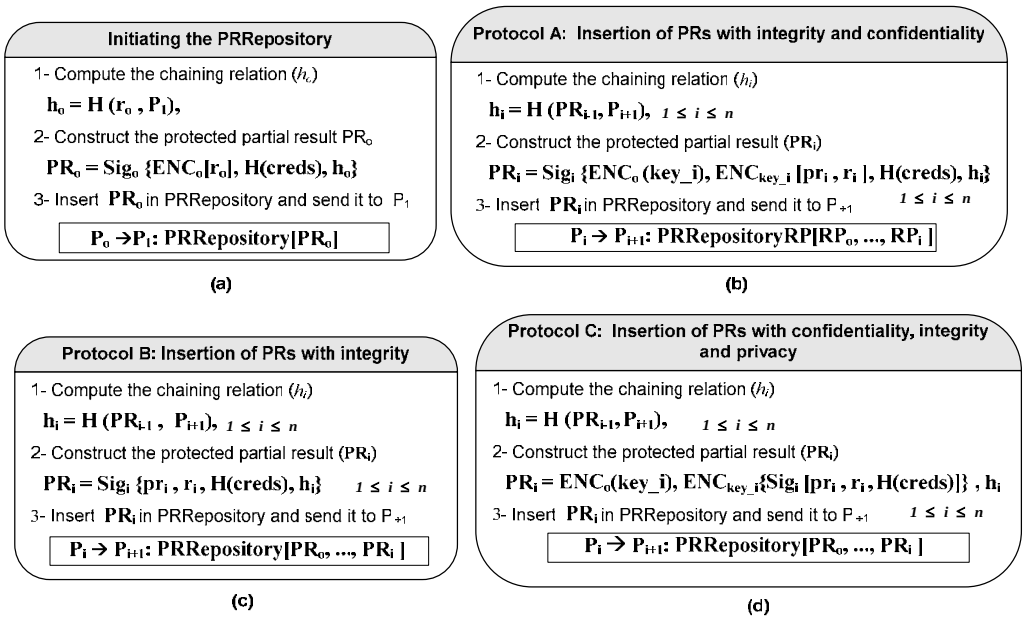

Figure 4- Partial Results Repository

The last step is what distinguishes the $MASS_{ma}$ protocol *A* from the protocol P1 from the KAG Family [8]. The modification consists in the inclusion of *H(creds)* – a cryptographic redundancy – which turns out to be appended to the partial result encapsulated in each platform. This modification aims to relate each of the collected partial results with the instance of the agent, thus, being able to overcome the attack described in [23][3], to which the protocols P1 and P2 from the KAG Family are susceptible. In addition, the hybrid cryptographic scheme used reduces the overhead caused by the public-key encryption of the partial results used in the KAG approach.

In protocol A, the encryption of data may impose a significant performance cost on the applications. This harm needs to be taken into consideration when the set of mechanisms to be employed in the protection of the agent is selected. Therefore, another protocol to the insertion of partial results - **Protocol B** - (Figure 4.c), which does not require the encryption of the partial results, is provided in $MASS_{ma}$.

[2] The temporary secret key is randomly generated by P_i and it is encrypted with the public key of the agent owner.
[3] In this attack, a fake agent can be created with the same state of a valid agent.

Protocol C is based on the protocol P2 from the KAG Family [8]. By changing the order of signing and encrypting a partial result, it is possible to hide the identity of the platforms that provided partial results while keeping the integrity assurances. However, the integrity of partial results cannot be verified in any visited platform. In the Protocol C (Fig. 4), step 1 is the same as in Protocol P2 from the KAG Family, while step 2 is different. It includes the cryptographic redundancy and it uses the hybrid cryptographic scheme.

2.2.5- Directed-Data Repository

This study also proposes a directed-data repository, called *DDRepository*, based on the technique implemented in the Ajanta platform [3, 16]. This technique allows a selective disclosure of an agent state, in which the agent programmer can implement a vector of objects (called *TargetedState*), in which each entry has specific target platforms [16]. The technique requires that the target platform be predetermined. As described in [23], the *TargetedState* technique has some vulnerabilities which allow data disclosure by malicious platforms. In attempting to overcome these vulnerabilities, this study proposes a refinement by means of the inclusion of a cryptographic redundancy which links each entry of the *DDRepository* to the instance of the corresponding mobile agent.

Figure 5 shows an example in which the agent programmer creates the *DDRepository* with two directed-data entries, to the P_1 and P_3 platforms, respectively. In order to ensure data confidentiality, the programmer, with the use of the public key of the target platform, must encrypt the temporary secret key created by the home platform and then encrypt, with the temporary secret key, the directed-data with the unique identifier of the agent instance. The home platform must also sign these data to ensure its integrity. When the agent is received in a target platform (platform *P1*), before proceeding to run the agent, the platform must verify the source of these data, their integrity, and whether or not are really associated with the received agent.

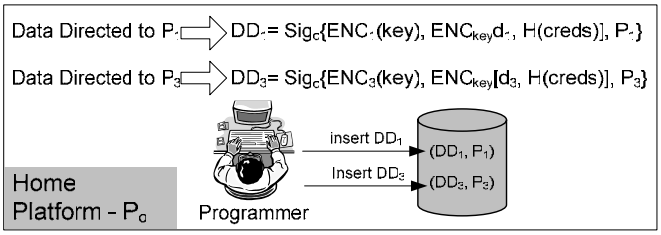

Figure 5- Inserting Directed Data in *DDRepository*

2.2.6- Mobile Agents Authentication

The *MASS* assists the programmer to build a protected mobile agent and also help the visited platforms to verify the agent integrity (code and repositories) by using a Multi-Hop Authenticator. By using this authenticator, it is possible to detect the integrity violation of the agent, and to provide information for identifying malicious platforms (social control mechanism). Figure 6 shows data repositories checking. According to steps 1 and 2 (Fig. 6), a platform must confirm, by means of code signature and the *RORepository* verifications, (1) that this agent has not been corrupted and (2) its association with a principal, its owner. In this way, modifications made by malicious platforms to the code and/or to the read-only data (*RORepository*) can be easily detected by any platform visited by the agent. In step 3, it is then necessary to verify the integrity of the data from the *DDRepository* and from the *PRRepository* which are being carried by the agent. Finally, in step 4, the *path register* object is verified and analyzed.

All the steps of the proposed authenticator are optional. The definition of which steps will be adopted for the authentication of a given agent will depend on the quality of protection (*QoP*) required by the owner of the agent and expressed in the agent's credentials. All the platforms visited must meet the minimum security requirements selected in the *QoP* attribute; otherwise, they will be considered malicious.

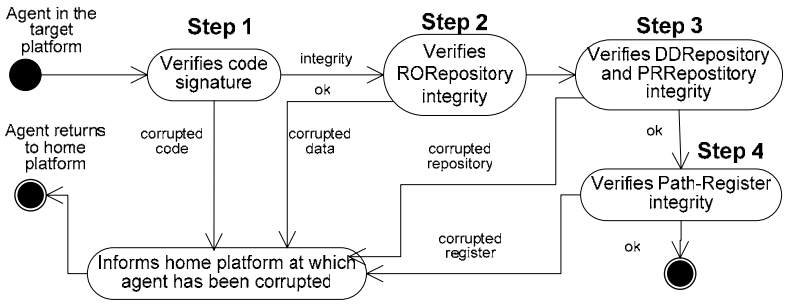

Figure 6- Multi-hop Authenticator

2.2.7 - Social Control Mechanisms

For the purpose of dealing with the occurrences of attacks against mobile agents, a mechanism of social control was defined and integrated into the proposed security scheme. This mechanism is based on platform reputation and aims to identify malicious platforms that are members of a SPKI Federation. Reputation is the image of the organization or entity, the way in which it is seen and recognized by those that interact with it, be the interaction direct or indirect [13]. The social control policy is used for isolating malicious platforms from the correct ones in the system. Therefore, the social control establishes a kind of

correct behavior that is imposed over the members of the federation and then each member supervises the behavior of the other members. There is no entity (either global or external to the group) that centralizes the social control.

Figure 7 shows the protocol for the identification of malicious platforms. The social control mechanism deals with two lists that indicate the reputations of the agent platforms belonging to a SPKI federation - the *black list* (identifying the malicious platforms) and the *red list* (identifying levels of reputation of the trustworthy platforms). The reputations indicated in the red lists are sorted according to the amount of notification occurrences of security violations.

Figure 7 also depicts an example of an *agent A* created and initialized in platform P_1 passing through platforms P_2, P_3 and P_4. Platform P_4 detects that the *RORepository* was corrupted and, thus, it suspends the execution of the agent and sends it back, together with the *RORepository*, to the home platform, which applies the protocol in order to locate the malicious platform. P_3 becomes suspect of the violation. In turn, P_4 is also considered suspect since it may be attempting to denigrate the reputation of P_3. Algorithm 1 below describes how the search for malicious platforms is performed.

Algorithm 1 SearchReputation (G, agent)

```
Require G = { i,...n; i ∈ N*} // Group of Federation Managers
1   if ( check_integrity (agent) = false) then
2     L ← ∅ {set of reputation lists}
3     while (G ≠ ∅) do
4       g ← getElement (G)
5       l ← getList(g)
6       L ← L ∪ l
7       G ← G \ g {Remove g from set G}
8     end while
9     verify (L)
10  end if
```

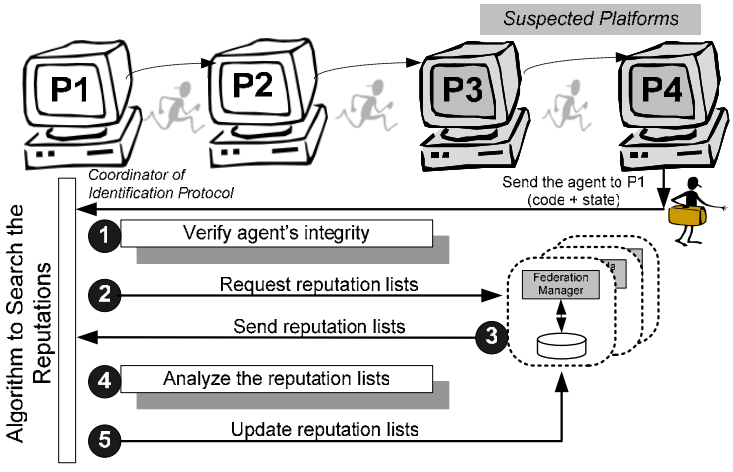

Figure 7- Identifying Malicious and Suspected Platforms

The message *SearchReputation*, performed by the home platform (coordinator of the protocol execution), is composed by the agent and the set of federation managers of the platforms visited by the corrupted agent. When the home platform receives the agent, it verifies the integrity of all data repositories checking for violation. When the violation is confirmed, a message is sent to the federation managers of the platforms visited by the corrupted agent, requesting the reputation lists (lines 3-8). Once the lists have been received, the coordinator must apply the social control policy. For example, if the reputation of one of the platforms is considered highly suspect, the coordinator must send to the corresponding federation manager of the suspect platform a notification about the security violation and also the identity of the platform under suspicion. This procedure will allow the latter to be included in the reputation *black list* in order to be, then, removed from this federation. If the reputation is considered moderately or slightly suspect, the coordinator must send to the federation manager a notification so that the reputation of this platform can be updated in the reputation *red list*.

3. Partners Selection and Search System in Collaborative Networks

The partner selection and search system, proposed in [19], is based on a hybrid agent architecture. This system exploits the benefits of the mobile agent paradigm in order to improve agility in the presentation of business opportunities to the companies and to achieve higher efficiency in the formation and analysis of possible virtual enterprises (VEs). Stationary agents, which represent every real company of a VBE (VE Breeding Environment), are responsible for interactions with the legacy systems of companies.

A prototype which implements the partners selection and search system - *MobiC-II* - was developed for the *TechMoldes* VBE, a group of mold makers in southern Brazil whose members have been collaborating to enhance their global competitiveness [19]. The main purpose of *Techmoldes* is to act either as a single or a larger productive entity in the market, combining the individual skills and resources of each member, but transparent to the final customer, however. When collaborating within *Techmoldes*, each member remains independent and autonomous at the same time, even to the extent of making business out of the VBE. Three classes of agents compose the *MobiC-II* system:

- **Broker Agents**: they are stationary agents responsible for receiving a business opportunity, distributing it to the potential enterprises, sending mobile agents to them, and collecting/electing the final VE composition.
- **Mobile Agents**: they are responsible for delivering a business opportunity to the enterprises, negotiating locally with them, and contacting other enterprises through the network, and, finally,

returning to the broker. These agents may have to perform different roles (missions) - from simple information messenger agents, to data researchers, to negotiators capable to make decisions and negotiations independently - without the need to wait on orders sent by the broker agent during the accomplishment of a task. Thereby, roles are created to the agents according to their desired function.

- **Enterprise Agents**: they are stationary agents that represent the enterprises and are responsible for receiving a business opportunity, evaluating it, accessing the local database to get the required information, and answering the business opportunity to the mobile agent.

The *MobiC-II* is shown in Fig. 8. Mobile agents are used as a means to travel through the selected enterprises in order to interact with the stationary agents to receive the required information (e.g., delivery time and capacity) or to negotiate lower costs or shorter delivery time.

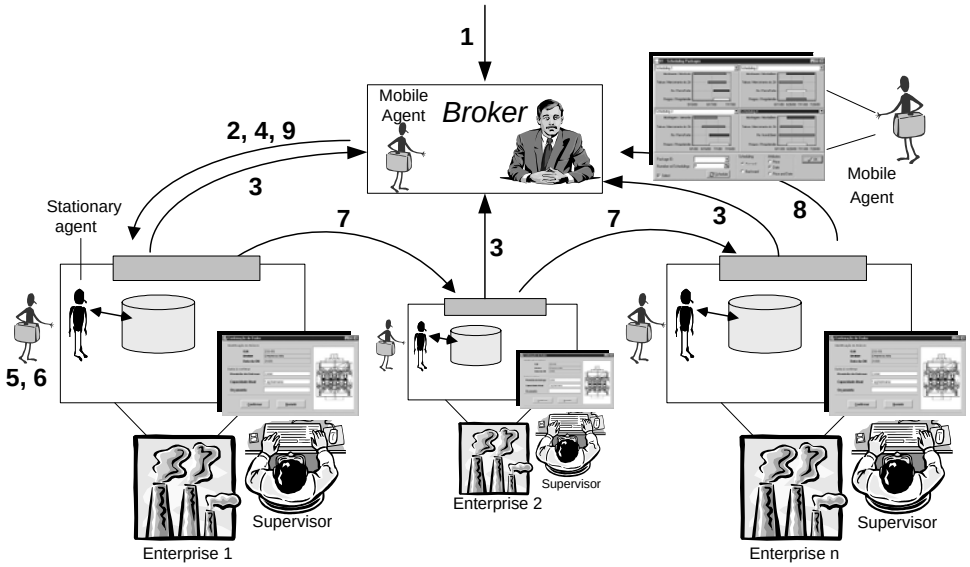

Figure 8- Scenario for the Partners Selection and Search System

When a business opportunity appears, it is received by the broker, which identifies (only) the potential enterprises that can supply each mold (**step 1**, Fig 8). A summary of the mold specification is immediately sent out to the enterprises through the *messenger agent* (**step 2**). Each enterprise receives it, evaluates its preliminary interest and capacity, and sends back an answer to the broker, either *yes* (expressing its interest) or *no* (**step 3**). The broker receives the answers and sends a mobile agent (*researcher* or *negotiator*) to the enterprises that answered *yes*. The agent is provided with the full business opportunity specification and the list of candidate enterprises it is supposed to visit (**step 4**). The mobile agent arrives at the first enterprise and interacts with the local stationary agent, asking for its delivery time and capacity (**step 5**). The local agent, acting as the enterprise's representative, retrieves this

information from its legacy system or local database. After that, the mobile agent asks the local supervisor about the price, as it is a very important piece of information in the molding sector. A negotiation process may be carried out locally (**step 6**). Then, with this piece of information, the mobile agent moves to the next enterprise from the list (**step 7**). This process is repeated until all the candidate enterprises from the list are visited. Finally, the mobile agent returns to the broker agent with the gathered proposals (**step 8**). The agent broker generates a set of possible VEs, assesses every VE composition and a human broker elects the most suitable one. Afterwards, the human broker sends a *win* or *lose* message to the enterprises (**step 9**). The *Techmoldes'* election criteria applied on this case are the lowest global cost and shortest delivery time. Finally, the broker assigns the responsibilities of each partner in the VE and then sends a *messenger* mobile agent in order to carry out the VE creation final message with the partners responsibilities and the VE contract (**step 10**).

The trust building process is indeed one of the most difficult issues to be overcome by the developers of VE solutions. The *MobiC-II* system considers the need of having more than one broker acting within a VE. This characteristic brings advantages to the *MobiC-II* due to the fact that (1) there is a reduction of a number of activities into a sole element and that (2) having many brokers makes a decentralized system's hierarchy possible, which aids the trust building process among participant enterprises. However, even though the members know one another and are aware of what they should do in order to be candidates for a given business opportunity, they get reluctant to share some kinds of information, such as prices, delivery dates and capacities. Some cultural, ethical, and managerial problems related to the use of the technology infrastructure have been pointed out as obstacles for a wider adoption of the VE paradigm by companies [11]. In collaborative networks, the adoption of security mechanisms that provide authenticity, confidentiality, integrity and non-repudiation, as the VE security policy specifies reinforced the trust building among the partners.

3.1- Threats in the Creation of Virtual Enterprises

In the steps of the MobiC-II system which use mobile agents (steps 2, 4, 5, 6, 7, 8, and 10 in Figure 8), all the security threats against the agent platform – *masquerading, denial of service, unauthorized access* and *repudiation* - are found in the *TechMoldes'* Scenario. For example, a malicious company can create a mobile agent that masquerades the actual agent, created by the Broker Agent, as an effort to gain access to sensitive information of competitive companies such as prices, delivery dates and capacities. This malicious agent can also launch attacks to consume an excessive amount of the agent platform's

computing resources; for example, it may request several fake budgets. Actions like these may prevent the platform from attending new business opportunities. In order to have access to the production capacities and sensitive information stored in database systems, the malicious agent may try to access the database through the Enterprise Agent.

Moreover, all security threats against the mobile agents are also found in the *MobiC-II* system – *masquerading, denial of service, eavesdropping* and *unauthorized access*. For example, with the purpose of impugning the election to select the most suitable VE, a malicious company may present a complaint for missing the business opportunity summary delivery - under the *messenger agent's* responsibility. A malicious platform may also introduce unacceptable delays to reply to an agent search, or even refuse to execute the agent, thus keeping the system from meeting the suitable VE partners. The threats against the communication channel, such as unauthorized modification, and eavesdropping, may compromise the dispatch of agents as well as the sending of messages among mobile agent platforms (found in steps 1, 2, 3, 7, 8 and 10 of the *MobiC-II* system, Figure 8).

3.2- Security Policy to the *TechMoldes* Scenario

An organization security policy is a set of rules and practices imposed by an organization to establish the operating limits of the users of a system, aiming at protecting the organization's sensitive data. During the design phase of the *MobiC-II* system, a security policy was defined to the *TechMoldes'* scenario [15]. This security policy determined which agents are mobile and which are stationary, the scope of the agents' functionalities, among others. A summary of the security policy rules is listed in Table 2.

3.3- Protecting Mobile Agent Systems to Improve the Trust Building Process

After identifying the threats and defining the organizational security policy, there was the analysis of both the security mechanisms supported by the *MASS* and the mechanisms needed to minimize or eliminate the exploitation of one or more vulnerabilities that would hinder trust building in the *MobiC-II* System. These mechanisms are listed in Table 3.

Rules	Description
R1	The integrity of read-only data carried by mobile agents should be provided by the MASS.
R2	Only the *TechMoldes* VBE's enterprises should have access to the summary of the mold specification and the full BO specification.
R3	Only the *TechMoldes* VBE's enterprises should take the broker's role and thus only these enterprises will be able to send (1) mobile agents with the BOs' specifications (messenger agents) as well as (2) researcher agents and (3) negotiator agents.
R4	The *MASS* must control the access of mobile agents to the platforms' sensitive data.
R5	The authenticity of the source of a mobile agent (its creator) must be verifiable.
R6	Only the *TechMoldes* VBE's enterprises may reply to a given BO (through the negotiator or researcher mobile agents). These collected proposals must be revealed only to the broker, and their integrity must be assured.
R7	The integrity and authenticity of the source of all messages exchanged between the agent platforms, while being sent by the communication channel, must be assured by the *MASS*.
R8	The integrity and authenticity of mobile agents, while being sent through the communication channel, must be assured by the MASS.
R9	Only a mobile agent owner may change its code.
R10	A VBE's enterprise may not deny that it has received a given BO if this really took place.
R11	An enterprise may not repudiate its own proposal in reply to a given BO.
R12	All VE partners may have access to the VE creation final message.
R13	All mobile agents should be sent only by VBE's enterprises.
R14	When a platform detects a violation of the agent integrity, it being of the code or state, the platform must immediately suspend the agent and inform the broker about the violation occurred. All platforms should supervise the platforms' behavior.

Table 2- Summary of the security policy rules to the *TechMoldes* Scenario

Security Mechanisms to the MobiC-II System	Rules satisfied
Repository of read-only data (*RORepository*) to protect the summary of the mold specification and the full BO specification	R1
Repository of Partial Results (*PRRepository* – Protocol A) to protect the proposals of Techmoldes companies	R6, R11
Repository of Directed Data (*DDRepository*) to protect the messenger agent sent by the broker to the VE partners (step 10, Figure 8)	R12
PathRegister object to store the itinerary of the agents (messenger, researcher, and negotiator agents)	R2, R10
Mobile agent authentication – integrity verification of repositories (RO, PR and DD)	R1, R6, R14
Mobile agent authentication - verification of the mobile agent signature	R3,R5, R9,R14
Procedures for Protection Domain Generation based on SPKI certificates	R4
Secure channel establishment – mutual authentication of agent platforms	R3
Secure channel establishment - use of SSL Protocol	R7, R8
Secure channel establishment - Reputation Verification of a Destination Platform	R13
Social Control Mechanism based on reputation applied in the *Techmoldes'* VBE	R14

Table 3- *MASS'* mechanisms in the *MobiC-II*

4. Implementation Results

A prototype of the security scheme - *MASS* - was implemented and integrated to the *MobiC-II* system in order to demonstrate its flexibility and suitability for distributed applications with mobile agents. The technologies adopted in this prototype favor the use of open standards and commercial-off-the-shelf (COTS) components to satisfy portability, scalability, and interoperability requirements.

As the **support layer of mobile agents,** we have chosen IBM Aglets[4], an open-source platform that uses Java as its mobile code language. The Aglets Software Development Kit (ASDK) provides mechanisms for code and state information mobility, and also for a computational environment. The **security scheme layer** of the prototype is composed of: (*i*) graphic user interfaces (GUIs) that aid the configuration and initialization of the security scheme; (*ii*) a library of classes, called *AgentSec*, which aids the agent programmer to build protected mobile agents, and (*iii*) objects that extend the functionalities of the Aglets platform in order to support the security scheme's mechanisms.

The **protocol for the secure channel establishment** and the multi-hop authenticator were implemented with the SDSI 2.0 library and with Java 2 cryptographic tools. The SSL support is provided by the iSaSiLk toolkit[5], and was integrated to the Aglets platform.

The MASS, through the ***AgentSec* library**, offers to the mobile agent programmer three secure repositories which can be combined to protect the integrity and confidentiality of the agent data. The objects in this library have been implemented for being independent from the agent platform. In order to aid the agent creation process and the use of secure data repositories, a GUI was implemented. This interface enables the owner to define which data repositories are going to be used and attached to the agent, with the resulting quality of protection (*QoP*). After selecting the mechanisms, the agent programmer has, as a result, a skeleton to the code of the agent (in the format of a *.java* file). With this generated structure and the *AgentSec* library, the programmer can easily continue the implementation of the mobile agent. When defining the quality of protection, the programmer must choose which protocol is to be used for the insertion of partial results, and also indicate whether the checking of this repository is going to be only in the home platform or in any other visited platform. It should be noted that, in the case of Protocol C, the repository can be verified only in the home platform. Figure 9 shows the implemented

[4] http://aglets.sourceforge.net/

[5] http://jce.iaik.tugraz.at/products/02_isasilk/

classes that compose the *AgentSec* library. The data repositories, when selected from a GUI, are created and defined in the `on_creation` method by any agent that expands the abstract class `com.ibm.aglet.Aglet`. Note that, during the creation process of a `RORepository`, the `Credentials` object associated to the agent is created and stored in this repository.

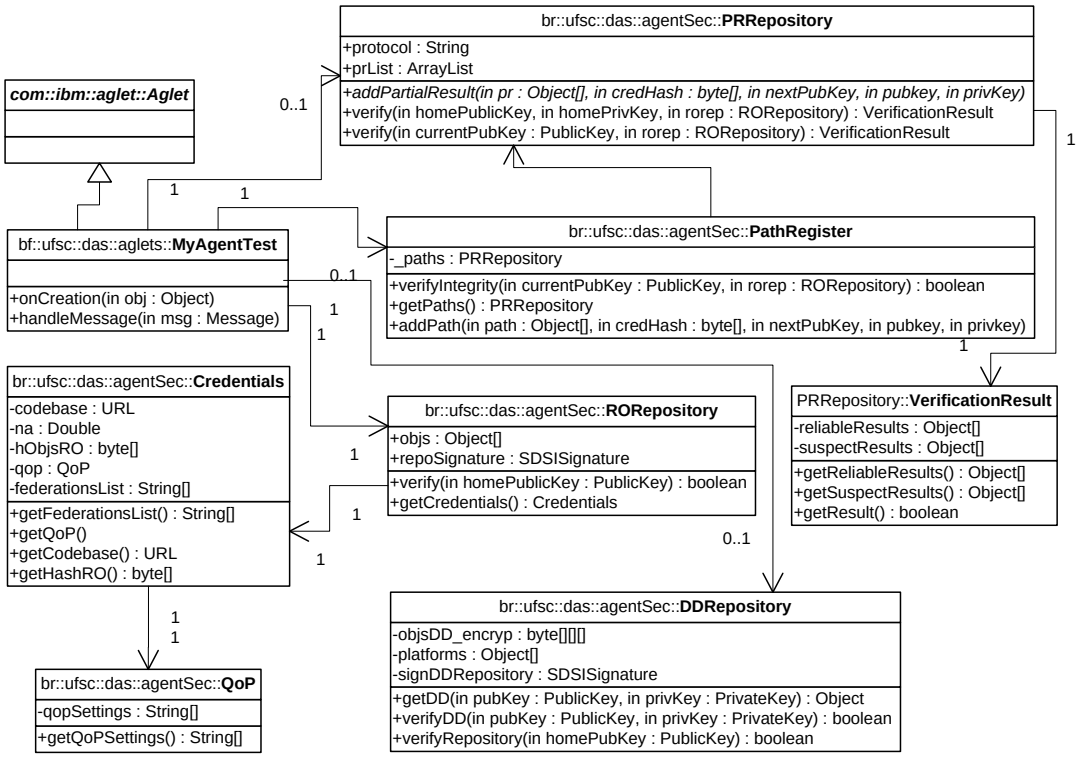

Figure 9- Class Diagram of the *AgentSec* Library

A mechanism for digitally signing the mobile agent code is also available in the *MASS*. As the implementation of this mechanism is directly related to the implementation of the Aglets Platform, the support to this mechanism is not offered through the *AgentSec* library. The decision on whether or not to use this mechanism is made by the programmer via a GUI of definition of the *QoP*. Here, the Aglets Platform has been adapted in order to optionally use the code signature mechanism, either when the agent carries the code or when the code is requested on demand.

The algorithm of the multi-hop authenticator was implemented as a stationary agent, called *SecurityInterceptor* (see Figure 9). This agent must be initiated in all platforms that are receiving the mobile agent, and has the role for intercepting the incoming mobile agent before its activation. This interception enables the verification of the incoming mobile agent integrity in the platform (mobile agent authentication). Only two levels of trust have been established: trustworthy or untrustworthy. An agent is considered trustworthy if all the algorithm steps were successful.

As the agent platform chosen for the prototype is based on Java, the secure interpretation of the agent code and the definition of the protection domains to mobile agents are provided, in part, by the Java 2 security model. The process for generating the set of permissions was defined to overcome the limitations related to the Java 2 access control model. There was the need for some extensions to the Java 2 security model so that the protection domain could be generated. The MASS' procedures, based on the agent privilege attributes (SPKI authorization chains) were implemented as defined in [14] and integrated into the Aglets platform.

As described in [19], in the *MobiC-II* system, agents were placed in two platforms. The mobile agents were coded in Java with the use of the Aglets platform. The stationary agents, coded in C++, were executed on MASSYVE-KIT platform[6]. CORBA is the technology applied to support the multi-platform interoperation.

4.1 Performance of the MASS' Prototype

This section presents some results of tests applied with the purpose of evaluating the performance of the $MASS_{ma}$ prototype, that is, evaluating the additional costs introduced by the use of secure data repositories and the authentication process of mobile agents. The tests were developed at LCMI/DAS, using a local network (Fast Ethernet - 100Mbps) and two dedicated computers with identical configurations - Pentium IV 2.4 GHz, 512 MB of RAM, whose operating system was GNU/Linux (kernel 2.4.21-199-atlon). Both computers had the Java 2 Software Development Kit (J2SDK), version 1.4.2-04.

4.1.1. Authentication of Mobile Agents that Using Secure Data Repositories

For this scenario, the following algorithms were used: *SHA1* and *RSA* (1024) for the digital signature, RSA (1024) for the key distribution, and 3DES for the data encryption. In the first experiment, a two-hop boomerang mobile agent was implemented with the use of a *RORepository*. When the agent is initiated, the read-only repository is created, a "Test" *String* is added to the repository, and the agent then migrates to the target platform. Before being run in the target platform, the agent is authenticated (the repository integrity is verified). The agent then returns to the home platform so that the repository integrity can be verified again. Table 4 shows the average time for an agent carrying a "Test" *String* without protection and for an agent carrying a *String* protected within the *RORepository*. Other measures

[6] http://www.gsigma-grucon.ufsc.br/massyve/mkit.htm

were taken so as to evaluate the sending and receiving times of this agent, which is now carrying a vector of bytes of varying size (256 bytes to 1 Mbyte; see Figure 10).

In order to evaluate the costs introduced by the employment of the data-directed repository - which uses symmetric encryption of data and asymmetric encryption of temporary keys - an agent carrying the *RORepository* and the *DDRepository*[7], with only a data input, was implemented.

Two-hop boomerang mobile agent with agent authentication	Time(ms)
Without *RORepository*	104.3
With *RORepository*	231.1
With *RORepository* and *DDRepository*	421.9

Table 4- Latency Comparison when an agent uses the *RORepository* and the *RORepository*

Table 4 shows the latency results of the mobile agent sending and receiving when the data input is a "Test" *String*. Figure 10 shows the results when the datum is a vector of bytes of varying size. Note that the results presented in Table 4 and in Figure 10 include the verification process of the agent (repositories checking). In this experiment, an expressive degradation of performance has been noticed. This is due to the data encryption that, in spite of being symmetric, causes a high cost in performance.

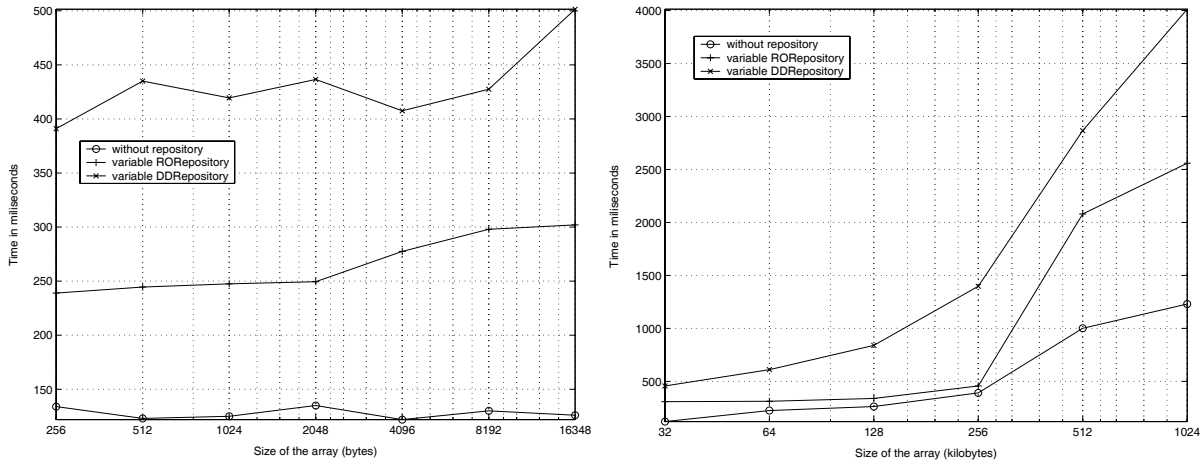

Figure 10- Comparative Graphs with and without the *RORepository* and *DDRepository* of Varying Sizes

In order to allow the conduction of the experiments with the *PRRepository*, there was the implementation of an agent and, consequently, the need for a more complex configuration of the agent platforms. Figure 11 illustrates the scheme of the *testPR* hops, in which six platforms are visited by the agent (three in each machine). The *testPR* agent departs from its home platform with a *RORepository*, and

[7] Whenever either a *DDRepository* or a *PRRepository* is attached to an agent, the *RORepository* must also be appended to it, since this is the one carrying the *Credentials* and the *QoP* needed for the verification of these repositories.

then starts, in turns, migrating from one machine to another, adding the partial results of each platform visited in the *RORepository*.

As Figure 11 shows, partial times are obtained in every two jumps until the total time is achieved, which is when the agent returns to the home platform. In each platform visited, the agent is authenticated. Table 5 compares the average times of the *testePR* agent itinerary, when the partial result added to each platform was a "Test" *String*".

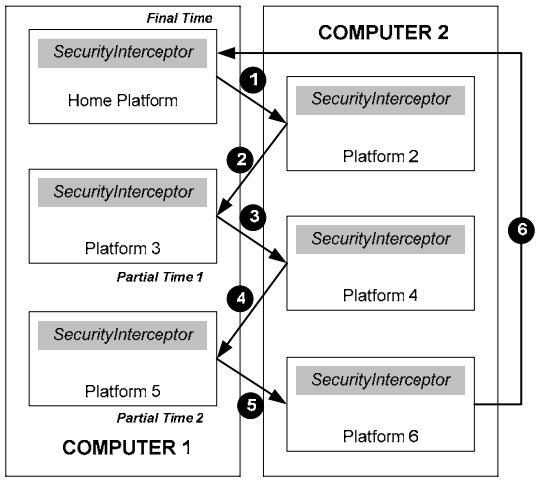

Figure 11- Scheme of hops to the *TestePR*

By analyzing the times presented in Table 5, it is possible to observe the cost introduced by the encryption processes, digital signature and computation of the chaining relation (*hash*), according to the configuration defined in each protocol in the $MASS_{ma}$. In the case of protocol A, the worst performance is justified for the additional time of the partial results encryption in this protocol. In this protocol, the previous partial results can be verified in any platform. The time increased in the last phase (the return to the home platform), since, at this point, five partial results were verified during the authentication process and the corresponding decryption of partial results also took place.

TestePR Agent (ms)	Without *PR*	Protocol A	Protocol B	Protocol C
Partial Time 1(hops 1 and 2)	120.8	861	775.5	803.1
Partial Time 2 (hops 3 and 4)	112.3	925.4	436.9	824.8
Partial Time 3 (hops 5 and 6)	107.3	2937.5	510.3	4072.6
Total (hops 1 to 6)	340.4	4723.9	1722.7	5700.5

Table 5- Latency Comparison when the agent uses the *PRRepository* and the *RORepository*

The analysis of the results obtained for protocol B led to the conclusion that the shortest times, if compared to protocol A, are due to the fact that there was no encryption for the partial results. If compared to protocol A, the times obtained through the use of protocol C, during the two first phases, are slightly shorter. The reason lies in the following fact: although this protocol applies encryption and digital

signature, it does not allow the results' integrity to be verified in intermediary platforms. In view of the fact that the integrity of the partial results inserted is only verified in the home platform, the time of the last phase increased considerably.

In a similar fashion as in *DDRepository* experiment, the test agent visits the specified platforms (see Figure 12) and collects the partial results which range from 256 bytes to 1 Mbytes. In other words, when each partial result collected has 256 bytes, as it returns to its home platform, the *PRRepository* will have five entries of 256 bytes (1280 bytes). During the execution of the experiment, it was concluded that an agent that carries a *PRRepository*, and uses either protocol A or C, is unable to finish its trip if the size of the data carried exceeds a given limit (see Figure 12). This is due to the fact that the checking process of the repository integrity needed more virtual memory than the amount made available by the computers used in these tests.

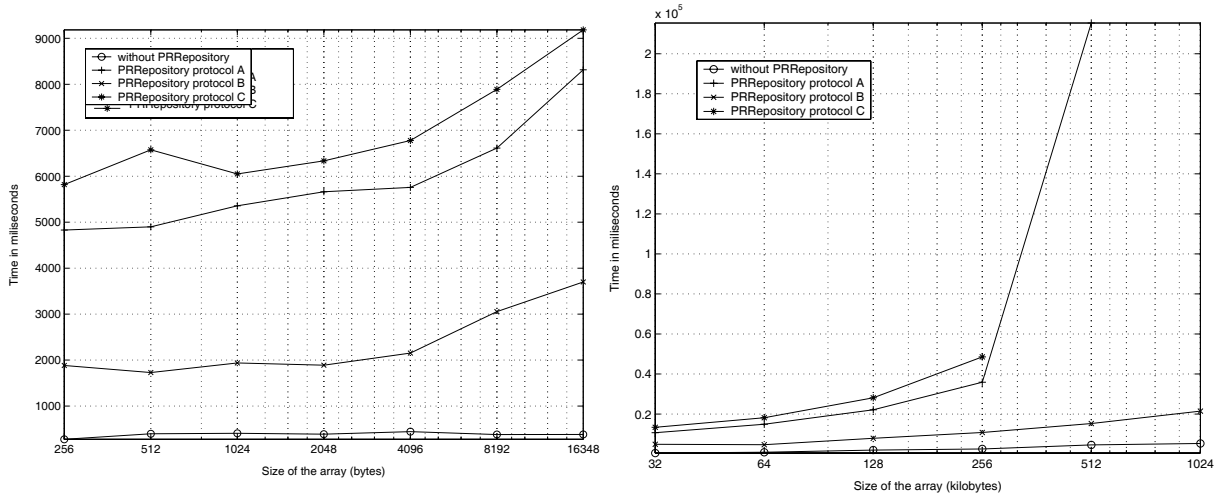

Figure 12- Comparative Graphs of an agent with and without a *PRRepository* of Varying Sizes

As far as protocol C is concerned, the limit was set when the agent collected a vector of 512 Kbytes in each platform and then arrived at the home platform, in an attempt to perform the last verification of the repository (with 2.5 Mbytes). It should be noted that, when the partial result was that of 1 Mbyte, the agent was only interrupted when it had already returned to its home platform, that is, when the *PRRepository* had 5 Mbytes of protected results (see Figure 12).

The above remarks point to the conclusion that the use of the encryption process for the partial results encapsulating (protocols A and C) causes the computational cost to rise significantly, mainly when the result collected is over 512 Kbytes and when more than four platforms insert results into this repository. For this reason, it is advisable (*i*) to adopt protocol B for the insertion of results in the

PRRepository, to large partial results (larger than 512 Kbytes) or (*ii*) to insert f results when protocols A and C are the ones being used.

Considering both the security requirements for the negotiator agent of the *MobiC-II* system and performance aspects, the conclusion was that the use of protocol A - which ensures the integrity and the confidentiality of the proposals encapsulated by the agent, and which allows the integrity to be verified by all of the VBE's enterprises - is the most suitable one for the *TechMoldes* scenario. As the proposals collected by the agent are XML documents ranging in size from 1 Kbyte to 4 Kbytes, and fewer than 15 platforms will present proposals, the computational cost, such as the use of the *PRRepository* with protocol A, corroborates the benefits brought by the use of the negotiator mobile agent.

5. Related Works

The analysis of the academic and commercial platforms of mobile agents described in the literature brings the realization that the problem of malicious platforms still lacks an appropriate solution in these systems. Only two platforms, SOMA [2] and Ajanta [3, 16], are committed to providing mechanisms for tackling this issue. Both platforms detect attacks originated from malicious platforms, but they do not prevent complex attacks of malicious platforms. Thus, in order to ensure the integrity of the agent code, these two platforms use digital signature techniques. In both of them, the non-mutable data of the agent are protected by means of a digital signature. However, only the Ajanta platform, with the Read-Only Container mechanism, is concerned with associating the agent's credentials with these data, and thus, creates a link between the read-only data and the agent. The *RORepository* proposed in this paper is based on the Ajanta platform mechanism. The difference lies in how the agent's credentials are defined and linked to the data. In the $MASS_{ma}$, the *H(creds)*, is built so as to distinguish the instances of the same agent. Therefore, a tamper-proof link between the instance of an agent and the *RORepository* is built; it is thus possible to detect when a malicious platform attempts to reuse an agent's credentials.

In order to protect the results encapsulated in the platforms visited by an agent, the SOMA platform uses detection techniques through TTP (Trusted Third Parties}) and MH (Multiple-Hop}) protocols. The first needs a trusted third party to encapsulate the results; this dependence is not desirable, however. In the MH protocol, as well as in the $MASS_{ma}$, the platform is responsible for the partial result encapsulation, and thus makes it difficult to change an intermediary result without changing the subsequent partial results. To accomplish this, a chaining relation must be cryptographically established among the results. The Ajanta platform, through the *Append-Only Container*, also employs the same

approach for the result encapsulation to ensure the integrity of partial results. Another study which follows this approach is the KAG protocols [8]. The main difference between these mechanisms lies in the way in which these chaining relations are cryptographically built and how these results are appended to the agent. In the protocols of the SOMA and Ajanta platforms, the integrity of partial results cannot be verifiable in any visited platform. In addition, the forward integrity property is not assured for these protocols, that is, it seems impossible to ensure that, when a partial result is modified, the results previously inserted can be entirely maintained. The mechanisms of both the SOMA and the Ajanta platforms were implemented, whereas the KAG Family protocols were not.

For the protocols for partial-result insertion supported in the $MASS_{ma}$, the main characteristics that contribute for a better result, if compared to the ones above-mentioned, are (1) the hybrid encryption scheme and (2) the cryptographic redundancy added during the computing of the protected partial results, which links the unique identifier of the agent's instance with each partial result collected in the visited platforms. This last characteristic ensures the scheme has a strong resilience against the insertion of partial results. As discussed in Section 4, all protocols were implemented and performance tests were executed aiming at measuring the performance degradation caused by the use of each protocol.

The works cited above do not offer mechanisms that automate the checking of an agent’s integrity when it is received in a platform. In the $MASS_{ma}$, the multi-hop authenticator has this role. Through this mechanism, it is possible for the platforms visited by the agent to guarantee the quality of protection (*QoP*), attributed to an agent by its owner. Another important contribution of the $MASS_{ma}$ is the prevention technique which avoids a mobile agent from being sent to a platform considered untrustworthy. With this technique, it is possible to help a mobile agent to establish the trust in a platform, based on the reputation of such a platform (which belongs to a federation).

Social control mechanisms can be combined to security schemes with the purpose of defining a correct behavior, which will be imposed over the participants of a group. Through the use of the infrastructure of SPKI service federations and with the aim of dealing with the occurrences of the attacks against mobile agents, a reputation-based social control mechanism is supported in the $MASS_{ma}$.

6. Concluding Remarks

Security issues still hamper the development of applications with mobile systems. Current security mechanisms do not present satisfactory results for protecting mobile agent platforms. There are even more limitations when we consider large-scale systems, which impose stronger requirements with

regard to flexibility and scalability. The MASS was motivated by the perception of these limitations and a concern about aspects of security specific to large-scale applications. This paper proposed an approach to improve trust building in Virtual Enterprises, especially in their creation phase (searching and selecting partners). In attempting to accomplish this, the MASS's security mechanisms were used for the conception of the MobiC-II system. In order to minimize the usual outweigh of these mechanisms, this work allows their configuration at the design phase of the application, using only the necessary mechanisms with their full features. The security mechanisms are used to protect, in open and large-scale systems, the communication channel, the agents platforms, and the agents themselves.

The work described in this article was fully implemented and its performance was properly measured and evaluated. Integration and adaptation of the MASS to the *MobiC-II* system was promoted in order to demonstrate its usefulness. To improve the reliability of the MASS' prototype, during its development, its correctness was analyzed by means of verification and validation activities (V&V) - software tests[8]. The implementation of the MASS and the evaluation of this implementation led to the conclusion that this scheme is adequate and useful for the protection of mobile agents in different kinds of application domain.

For continuing this work, a study is being prepared about how to protect confidentiality of the agent code using some mechanisms that transform a program into an equivalent one that is harder to reverse engineering and so minimizing threats against the agent's intellectual property. Also, the implementation of the control social mechanism based on SPKI Federation concept is one of next goals in this project. Finally, the use of a test methodology to evaluate the security risks of prototype's mechanisms is being prepared.

Acknowledgements

This work has been developed within the scope of the Brazilian IFM (www.ifm.org.br) and European FP6-IP ECOLEAD (www.ecolead.org) projects. The authors would like to thank the financial support received and the members of these projects for their contributions.

[8] Two types of tests were applied: unit tests and systems tests.

References

[1] A. Santin A, J. Fraga, F. Siqueira, E. Mello. Federation WEB: A scheme to compound authorization chains on large-scale distributed systems. 22nd Symposium on Reliable Distributed Systems, 2003.

[2] A. Corradi, M. Cremonini, R. Montanari, C. Stefanelli, Mobile agents integrity for electronic commerce applications. Information Systems vol. 24, pp. 519-533, 1999.

[3] A. Tripathi, T. Ahmed, N. Karnik, Experiences and future challenges in mobile agent programming. Microprocessors and Microsystems (2001) 121-129.

[4] C. Elisson, SPKI Requirements (RFC 2692). The Internet Engineering Task Force. (1999) http://www.ietf.org/rfc/rfc2692.txt.

[5] D. Chess, B. Grosof, C. Harrison, D. Levine, C. Parris, G. Tsudik. Itinerant agents for mobile computing. IEEE Personal Communications, 2(5):34–49, (1995).

[6] F. B. Schneider. Towards fault-tolerant and secure agency. In M. Mavronicolas e P. Tsigas, editors, 11th International Workshop on Distributed Algorithms (WDAG'97), volume Lecture Notes in Computer Science of LNCS, pp. 1–14. Springer, 1997.

[7] F. Hohl. Time limited blackbox security: Protecting mobile agents from malicious hosts. In Giovanni Vigna, editor, Mobile Agents and Security, volume 1419 of LNCS, pp. 92–113. Springer, 1998.

[8] G. Karjoth, N. Asokan, C.Glüc: Protecting the computing results of free-roaming agents. In: Proc. of the Second International Workshop on Mobile Agents, 1998.

[9] G. Necula, P. Lee. Safe, untrusted agents using proof-carrying code. In Giovanni Vigna, editor, Mobile Agents and Security, volume 1419 of LNCS, pp. 61–91. Springer, 1998.

[10] G. Vigna.: Cryptographic traces for mobile agents. In Vigna, G., ed.: Mobile Agents and Security. Volume 1419 of LNCS, pp. 137-153, Springer, 1998.

[11] L. M. Camarinha-Matos, H. Afsarmanesh. Dynamic virtual organizations, or not so dynamic ? In: Third IFIP Working Conference on Virtual Enterprise, pp. 111-124, 2002.

[12] L. M. Camarinha, H. Afsarmanesh. The emerging discipline of collaborative networks. In Virtual Enterprises and Collaborative Networks, Kluwer Academic Publishers, ISBN 1-4020-8138-3, IFIP Vol. 149, Aug 2004.

[13] L. Rasmusson, A. Rasmusson, A, S. Jansson. Using agents to secure the internet marketplace - reactive security and social control. In: Proc. 2nd Int. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology, 1997.

[14] M. S. Wangham, J. S. Fraga, R.R. Obelheiro., G. Jung, E. Fernandes, Security mechanisms for mobile agent platforms based on spki/sdsi chains of trust. In C.L. et al., ed.: Software Engineering for Multi-Agent System II. Vol. 2940 of LNCS. Springer, 207-224, 2004.

[15] M. S. Wangham, J.S. Fraga, R. Schmidt, R. J. Rabelo, MASS: A Mobile Agent Security Scheme for the Creation of Virtual Enterprises. Proceeding of Mobility Aware Technologies and Applications. Vol. 3284 of LNCS. Springer pp. 234-243, 2004.

[16] N. Karnik: Security in Mobile Agent System. PhD thesis, University of Minnesota, 1998.

[17] OMG: Mobile agent facility specification. OMG Document 2000-01-02, 2000.

[18] O. Esparza, M. Fernandez, M. Soriano, J. Muñoz, J. Forné. Mobile Agent Watermarking and Fingerprinting: Tracing Malicious Hosts, DEXA 2003, Vol. 2736 of LNCS. Springer, 2003.

[19] R. Rabelo, M. S. Wangham, R. Schmidt, J. Fraga: Trust building in the creation of virtual enterprises in mobile agent-based architectures. In: Processes and Foundations for Virtual Organizations, Kluwer Academic Publishers, IFIP, 2003, pp. 65-72.

[20] R. Rabelo, F. Baldo, R. Tramontin Jr, A. Klen, E. Klen. Smart Configuration of Dynamic Virtual Enterprises, In Virtual Enterprises and Collaborative Networks, Ed. L. M. Camarinha-Matos, Kluwer Academic Publishers, 2004. pp. 193-204.

[21] Sun. Java 2 sdk. v1.4 security documentation, February 2002. http://www.java.sun.com/security/index.html.

[22] T. Sander, C. Tschudin. Protecting mobile agens against malicious hosts. In Vigna, G., ed.: Mobile Agents and Security. Volume 1419 of LNCS. Springer, 1998.

[23] V. Roth. On the robustness of some cryptographic protocols for mobile agent protection. In Picco, G.P., ed.: Mobile Agents. Volume 2240 of LNCS, pp 1-14. Springer, 2001

[24] W. Farmer, J. Guttman, V. Swarup, Security for mobile agents: Issues and requirements. In: Proc. 19th National Information System Security Conference, 1996.

[25] W. Farmer, J. Guttman, V. Swarup. Security for mobile agents: Authentication and state appraisal. In 4th European Symposium on Research in Computer Security (ESORICS'96), 1996.

[26] W. Jansen, T. Karygiannis. Mobile agent security. Technical Report NIST Special Publication 800-19, National Institute of Standards and Technology (1999).

Authors' Biographical Notes

Michelle S. Wangham is a postdoctoral research associate in Department of Automation and Systems at the Federal University of Santa Catarina. His main interest areas of research include Security in Distributed Systems, Mobile Code and Collaborative Networks. She is currently involved in the European FP6 IP ECOLEAD project and in two Brazilian projects as a researcher. In 2004, Michelle received a PhD in Information Systems from Federal University of Santa Catarina - Brazil.

Joni da Silva Fraga received the B.S. degree in Electrical Engineering in 1975 from University of Rio Grande do Sul (UFRGS), the MSE degree in Electrical Engineering in 1979 from the University of Santa Catarina (UFSC), and the PhD degree in Computing Science (Docteur de l'INPT/LAAS) from the Institut National Polytechnique de Toulouse / Laboratoire d'Automatique et d'Analyse des Systèmes, France, in 1985. Also, he was a visiting researcher at UCI (University of California, Irvine) in 1992-1993. Since 1977 he has been employed as a Research Associate and later as a Professor in the Department of Automation and Systems at UFSC, in Brazil. His research interests are centered on Security, Fault Tolerance and Distributed Systems. He has over 95 scientific publications and is a Member of the IEEE Computer Society and of Brazilian scientific societies.

Ricardo José Rabelo took his Ph.D. in Robotics and Integrated Manufacturing at New University of Lisbon, Portugal, in 1997. He is Associated Professor of the Department of Automation and Systems at the Federal University of Santa Catarina (UFSC) since 2000. He is co-founder of the G-SIGMA – Intelligent Manufacturing Systems Group (www.gsigma.ufsc.br). He worked as consultant for 3 years (1983-1985) and in the Mercedes-Benz of Brazil (1986-1987), leading a division of software development for the Computer Aided Manufacturing area. His main current areas of research include: virtual organizations, systems and information integration/interoperation, multi-agent systems, shop-floor supervision and decision support systems. He has participated in several national/Brazilian (e.g. IFM, Manet) and international research projects (ESPRIT and IST programmes) as well as in cooperation projects with Europe (ECLA, CYTED, INCO and KIT programmes) - most of them as the UFSC representative / technical contact. He has more than 90 publications, including conference proceedings, journals and book chapters. He is the Brazilian representative in the IFIP WG 5.3 and 5.5 subgroups, consultant ad hoc of the Capes agency for applied research and other governmental institutions. Besides

that, he has been involved in the organization and program committees of several national and international conferences, such as of PRO-VE.

Lau Cheuk Lung is an associated professor in the Department of Computer Science at Pontifical Catholic University of Paraná - Brazil, where he has been working since 2003. Currently, he is conducting research in fault tolerance, security in distributed systems, distributed Algorithms and middleware. From 1997 to 1998, he was an associate research fellow at University of Texas at Austin, working on the Nile Project. From 2001 to 2002, he was a postdoctoral research associate in the Computer Science Department at University of Lisbon, Portugal. In 2001, Lau received a PhD from Federal University of Santa Catarina - Brazil.