SPIG
Security by Privileges, Inspection and Guaranty for Ambient Agents Group

Nardjes Bouchemal[1,2] and Ramdane Maamri[1]
[1]*LIRE Laboratory, Computer Science Department, Mentouri University of Constantine, Constantine, Algeria*
[2] *University Centre of Mila, Mila, Algeria*

Keywords: Ambient Intelligence, Ambient Agents Group, Security, Inspection, Guaranty.

Abstract: In a future vision of Ambient Intelligence – or AmI – our surrounding environment will integrate a pervasive and interconnected network of devices equipped with sensors, actuators and ambient agents. However, it is a big security issue when building this kind of complex systems, and it is not enough that ambient agent platform provides a set of standard security mechanisms such as sandboxing, encryption and digital signatures. Furthermore, we must take into account ambient agent limitations and we mustn't endow it with complex cryptography concepts or historic data. This is why our approach, proposed in this paper, is based on cooperation between group members and behavior inspection of new integrated agents, in addition of some cryptography concepts. The proposed protocol is based on cooperation, collective decision, guarantee and attribution of privileges according to the trust degree of agents.

1 INTRODUCTION

The first vision about pervasive computing was proposed by Mark Weiser in his famous article "The Computer for the 21st Century" 1991 (Weiser, 1991). His vision of computers fully integrated in the human environment and gracefully providing information and services to users is still an open issue in computer science and computer engineering.

The ambient intelligence paradigm builds upon pervasive computing, ubiquitous computing, profiling practices, context awareness, and human-centric computer interaction (Augusto, 2009), (Weber, 2005).

Consequently, the development of AmI environment needs robust technologies, such as Agents (Ferber, 1995), that respond to AmI characteristics.

They have already been successfully applied to several AmI scenarios such as education, culture, entertainment, medical domain, robotics and home (Tapia, 2010). In these systems, agents are called ambient agents.

However, developing agents in complex AmI environments leverages the common issues of distributed applications. It also poses new security challenges to handle the dynamicity of such environments. Indeed, ambient agents will operate on portable electronic devices with little autonomy energy, limited computing and storage capabilities that vary from one device to another (Ko and Ramos, 2009).

Thus, ambient agents groups are at risk if there is no control of different participants, or agents who require communicating. The crucial question is how to ensure any confidentiality, integrity and privacy of exchanged messages and data if, at the outset, we are not sure that we communicate with the correct entity? (Stajano, 2010)

This article presents an approach to protect ambient agents groups by taking in to consideration the new challenges required by ambient intelligence environment.

In section two we sketch out security approaches for classic multi agent systems and we discuss their insufficiency to AmI field. In section three we define the protocol, based on guaranty, inspection of new integrated agents, on cryptography concepts and attribution of privileges according to the trust degree of integrated agent. Section four presents implementation issues using JADE platform and summarizes preliminary comparative results. Finally, a conclusion recaps the paper and future works.

DOI: 10.5220/0004261703180323
In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence* (ICAART-2013), pages 318-323
ISBN: 978-989-8565-39-6

2 RELATED WORK

2.1 Multi Agents System Protection

The goal of this section is to present several viable technologies that can be used in multi-agent systems in order to provide a secure infrastructure.

Then we discuss the robustness of these approaches to respond to different requirements of ambient intelligence.

The Protected Computing approach is based on the partitioning of the software elements into two or more parts. Some of these parts (called private parts) are executed in a secure processor, while others (public parts) are executed in any processor even if it is not trusted.

In general, the Protected Computing model requires the use of secure coprocessors that have asymmetric cryptography capabilities; secure storage to contain a key pair generated inside the coprocessor and ensure that the private key never leaves it (Maña 2006).

Trusted Computing Platforms (TCG, 2005) take the advantage of the use of a hardware element in order to provide a secure environment. These hardware elements are called TPM (Trusted Platform Modules). A TPM is a microprocessor with some special security features. The key idea behind the Trusted Computing Platform is to build a trusted environment starting from the TPM and extending the trust to the rest of the system elements. At the beginning of the execution the only trusted element in the system is the TPM.

The technique called proof-carrying code (PCC) (Necula, 1997) is a general mechanism for verifying that a software element can be executed in a secure way. For this purpose, every code fragment includes a detailed proof called code certificate.

Maña et al. in (Maña 2006), proposed an approach to protect a society of collaborating agents, by making every agent collaborate with one or more remote agents running in different hosts. These agents act as secure coprocessors for the first one. Likewise, these agents are in turn protected by other agents.

The Static Mutual Protection strategy can be successfully applied to many different scenarios. However, there will be scenarios where (i) it is not possible to foresee the possible interactions between the agents at development time, (ii) where the agents are generated by different parts.

Maña et al. in (Maña 2004) and (Maña 2007) proposed a new strategy called Dynamic Mutual Protection where each agent is able to execute arbitrary code sections on behalf of other agents in the society. Each agent includes a public part, an encrypted private part and a specific virtual machine.

2.2 Discussion

Ambient agents have limited capabilities due to their closely attachment with the device capabilities. They are cognitive, but hold little knowledge, don't require much resources and form very simple plans especially when they are embedded in very small devices like sensors.

Furthermore, in the context of ambient intelligence, entities are intelligent and can pretend being confident just to be integrated to the ambient agents group. Then, they can make dangerous actions and may even lead to the destruction of the group.

Consequently, the question to ask is: *can presented approaches respond to ambient agents properties?*

Previous presented works don't inspect agents after integration; they just verify the origin and the authenticity. But even the origin is trustworthy; agent can be altered during its displacements.

Furthermore, presented approaches use complex cryptographic concepts, which make them robust, but more difficult to realize in case of limited ambient agents' resources. Moreover, these approaches are limited by the number of agents included in the group, where the management becomes more and more difficult.

In conclusion, during the design of ambient agent's security, we must take in mind ambient agents' limitations and we mustn't endow them with complex cryptographic concepts or historic data. This is why our approach is based on cooperation, collective decision and division of security tasks between agents of the group.

3 SPIG: SECURITY BY PRIVILEGES, INSPECTION AND GUARANTY

We proposed in previous works (Bouchemal, 2012) an approach based on guaranty, where an agent is integrated into a group only if an agent from the group knows it. The problem is posed when no agent knows the new one. In this section, we bring ameliorations and consider the case where an agent is completely unknown.

3.1 Inspiration

In the design of the protocol, inspiration was taken from the human behaviour and thinking. In order to more understand the idea, we present in this section a real life scenario. Let's consider a home where lives a couple and their children: Charlie and Betty, having consecutively fifteen and thirteen years old. The father works in a computer company with other persons chaired by a director. On the other hand, Charlie and his sister attend a school, near their house, managed by a principal. They have several friends and teachers within the school.

One day, the door rings, the father opens. He finds a teenager claiming to be Charlie's friend. The father calls his son and asks him. Charlie confirms that he knows the visitor. The father brings Charlie's friend, but he is suspicious, so he restricts visitor displacements in the house and recommends to Betty and her mother to keep an eye.

Another day, the father invites some colleagues to have lunch in the home. After that, he takes them to visit the garden and to drink coffee there. They are supervised just by the father (eventually by the mother) because these gents are supposed trustworthy.

3.2 Departure Points

We consider a group of ambient agents embedded in mobile devices and represent an object and/or a user. We define some departure points in what follows:

- Each member of the group has a set of characteristics:
  - *Id_Agent:* A unique identification within the group.
  - *Priv:* Privileges within the group (set of permitted actions).
  - *LiD:* The list of identifications agents of the group.
  - *TrustDeg:* A degree of trust within the group: doubtful, malicious or trust.
  - *ResidTime:* The estimated residence time within the group.
  - *DevRep:* Representation of the device resources: energy, memory and processor capabilities.

- Each group has a specific and trust agent called ***representative,*** having maximum privileges, powerful device and with longer time of residence.
- Representative of the group has the instantly list of all group agents with their priviliges, trust degrees and behavior reports.
- Members of the same group know each other and cooperate to complete various tasks. They share a Common Public Key (CPK) known only by group members and frequently changed by the representative.
- Members have communication keys (Cki for an agent i) to establish secure channels with the representative. Cki is frequently changed.
- The group have two vice representatives having most privileges, residence time and powerful device. The goal is to ensure the suitable conduct of the group in case of attacks against the representative.
- An agent can belong to two groups at once, or more (Figure.1). In this case it *knows* agents and have keys of both groups.

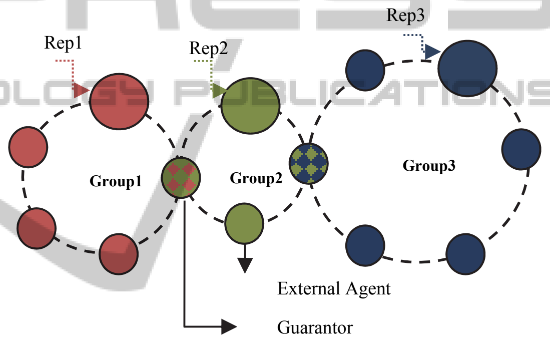


Figure 1: Agents belonging to more than one group.

3.3 The Protocol

The key idea is based on behaviour inspection of new integrated agents, cryptography concepts, guarantee and attribution of privileges.

When an external agent requests communication or data from the group, it contacts either a member or the representative. If it contacts a member, this one redirects the request to the representative. If the external agent contacts directly the representative, it cooperates with the group's agents to verification, integration or rejection of the request, and inspection after integration.

3.3.1 Verification

When the representative group receives an integration demand from an external agent, it sends its identity to group members. We study two cases before responding: First, external agent is known by at least one agent from the group (so called The guarantor), when both agent belong to another group (Group 2 in Figure.1).

The guarantor sends a confirmation to representative who will add the external agent to the group. The integration is done differently depending on the privileges of the guarantor.

In the second case, external agent is unidentified by any agent of the group. Representative delegates a committee to examine it. The committee is a subset of agents with most privileges, resources, residence time and trust degree.

Committee members use sandbox technique (Gong, 1997) by sending different trivial code and data to the external agent. After execution, it resends results, code and data. Agents of the committee check the code, data and results and send their reports including attributed trust degree and privileges to representative who makes the last decision. If the trust degree is malicious, the request is rejected. Else, if the degree is doubtful, the request is accepted but with minimum privilіges and the agent is inspected. If the degree is trust, the request is accepted and the agent is integrated, but must also be inspected.

3.3.2 Integration

The new agent integration within the group is done only if trust degree is not malicious. First, representative establishes a secure channel with the new agent by sending a communication key Ck'. Then it sends the common public key: Ck'(CPK), and the identities list of group members (LiD). It also sends new agent identity, trust degree and privileges to members. After that, the representative informs all members that the integrated agent must be inspected.

3.3.3 Inspection after Integration

Members cooperate to keep track of new integrated agents in order to confirm their intentions. They oversee the behavior of integrated agent and they make a report to the representative, if they discover malicious or doubtful behavior. The representative can make changes in status, and privileges (Figure.2).

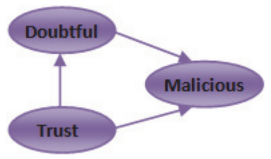


Figure 2: Possible changes of integrated Agent's status.

It can revoke agents having malicious behavior by changing CPK and adding it in its black liste. After any changes, the representative informs all group members.

4 IMPLEMENTATION AND EXPERIMENTATION

To validate the proposed protocol, we conduct, in section 4.1, an implementation using JADE platform (Java Agent DEvelopment framework) (Bellifemine, 2004), (Caire, 2007). Jade is Java-based and FIPA (Foundation for Intelligent Physical Agents) compliant development framework. In section 4.2, we establish a comparison between our protocol and the Protection Computing Approach.

4.1 Implementation of Groups and Agents

Agents have been designed so that they are simple, flexible, and so that an agent with the same structure can run both on a simple processor assigned to a sensor and on a powerful computer. The agents are cognitive, but hold little knowledge and form very simple plans.

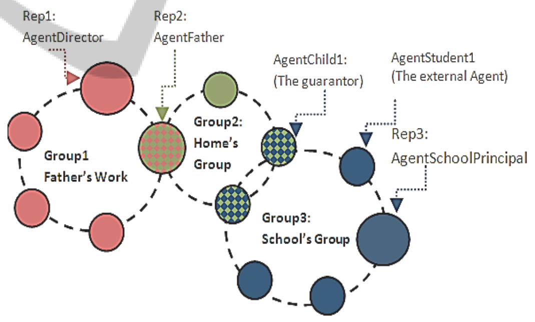
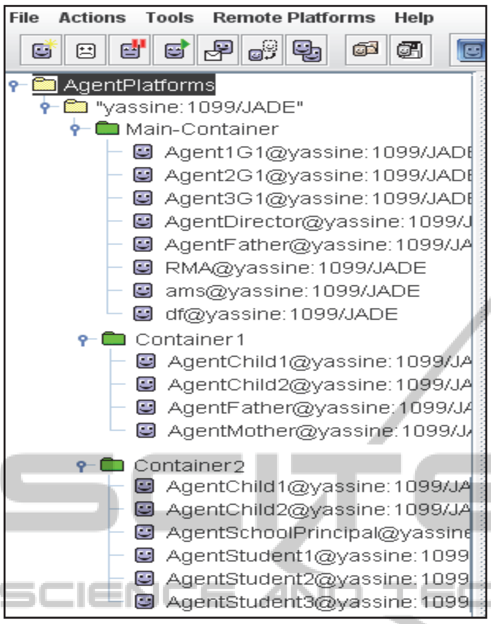


Figure 3:Agents and Groups.

We would like to simulate the scenario presented in section 3.1. Thus, we create three containers each one represents a group: main container for group1, modelling the father's work group and comprises *AgentDirector* (the representative of the group), *AgentFather*, Agent1G1, Agent2G1 and Agent3G1 (Figures.3 and Figure4). Container1 represents Charlie's home group and includes *AgentFather* (the representative of the group), *AgentMother*, AgentChild1 (for Charlie) and AgentChild2 for his sister. Finally, container3 represents Charlie's school group, icluding: *AgentScholPrincipal* (the representative), *AgentChild1*, *AgentStudent1*, *AgentStudent2* and *AgentStudent3*.

Agents are embedded in various devices (laptops for the representatives, PDA and smart phones for the rest of agents).



```
AgentPlatforms
  "yassine:1099/JADE"
    Main-Container
      Agent1G1@yassine:1099/JADE
      Agent2G1@yassine:1099/JADE
      Agent3G1@yassine:1099/JADE
      AgentDirector@yassine:1099/J
      AgentFather@yassine:1099/JA
      RMA@yassine:1099/JADE
      ams@yassine:1099/JADE
      df@yassine:1099/JADE
    Container1
      AgentChild1@yassine:1099/JA
      AgentChild2@yassine:1099/JA
      AgentFather@yassine:1099/JA
      AgentMother@yassine:1099/J
    Container2
      AgentChild1@yassine:1099/JA
      AgentChild2@yassine:1099/JA
      AgentSchoolPrincipal@yassine
      AgentStudent1@yassine:1099
      AgentStudent2@yassine:1099
      AgentStudent3@yassine:1099
```

Figure 4: Agents and containers using JADE.

To implement agent's behaviours in JADE, we use behaviour composition by building simple behaviours. The basis for this feature is provided by the *CompositeBehaviour* class included in the *jade.core.behaviours package.*
Messages exchanged by agents are instances of the *jade.lang.acl* from *ACLMessage* class, which represents ACL messages. It contains a set of attributes as defined by the FIPA specifications.

4.2 Experimental Scenes

To verify the efficiency of the proposed protocol, we advise experimental scenes using SPIG protocol and the Protected Computing approach.

As mentioned in section 2.1, the Protected Computing Approach uses a trusted processor to enforce the correct execution of the private parts of the program.

Therefore, using the protected computing model, the code of each agent is divided into public and private parts. For reasons of simplicity, we will consider that the code of each agent is divided in two parts: a public one and a protected one.

4.2.1 Scene One: The External Agent is Known

AgentStudent1 from *Container2* requests integration into *Container1*; it contacts *AgentFather* (the representative). *AgentChild1* knows *AgentStudent1*, so it can be integrated with same *AgentChild1's* privileges, and must be inspected by all members: *AgentFather*, *AgentMother* and *AgentChild2*.

4.2.2 Scene Two: Malicious Action from AgentStudent1

After integration, *AgentStudent1* becomes untrusted. We simulate this behavior by injecting a malicious code into *AgentMother,* which discovers this action and contacts *AgentFather* to revoke it.

4.3 Experimental Results

In scene one, we have calculated integration time of external agent, when it is known, relation to the number of agents in the group. We have established comparison graph, as shown in Figure.5.

We can observe that the latency taken by the cryptographic operations, division of agent's code and the communication with the coprocessor, is an important factor that affects the performance of the protected computing approach.

In SPIG protocol, integration time is not so considerable, because when the agent is known, it is integrated faster even the number of agents is big.

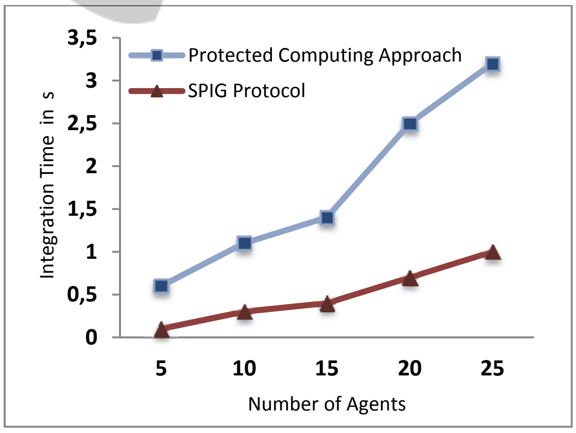


Figure 5: Comparison of agent's integration time.

Concerning scene two, we calculate necessary time to detect AgentStudent1 attack using SPIG protocol then the protected computing approach relation to increasing number of integrated agents.

For protected computing approach, when the number of agents increases, the management becomes more difficult, principally because of devices resources' limitation, rising of divided parts of agents and the overflow of trusted processor (Figure.6). These circumstances make the verification of malicious actions slower.

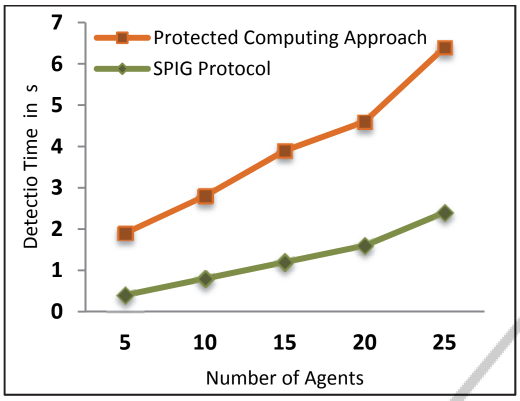


Figure 6: Comparison of attack detection time.

5 CONCLUSIONS

We presented in this paper an approach to protect ambient agents group, where each agent is embedded to a device and represents a user.

If a new agent wants to communicate and share data with agents of this group, it must be verified, integrated if it is not malicious, and inspected after integration. These functions are done cooperatively by the members.

We have detailed the protocol and introduced a set of encryption keys: CPK (Common Public Key) shared and known by all members of a group, Cki (Communication Key for Agent i) sent by the representative to an agent i. We presented a simple case study in health domain, and finally we presented some implementation issues using Jade platform. Our on-going work is to apply the proposed approach in a real case and real devices, by using a professional AmI platform. We would like to test the reliability in large scale.

REFERENCES

Augusto, J. C., 2008. Ambient Intelligence: Basic Concepts and Applications. In *Computer and Information Science: Software and Data Technologies*, Volume 10, Part 1. Berlin, Germany: Springer Berlin Heidelberg.

Bellifemine, F., Caire, G., Trucco, T., Rimassa, G., 2004 *Jade Programmer's Guide*, version 3.2 . *The book.*

Bouchemal, N., Maamri, R., 2012. SGP: Security by Guaranty Protocol, *DCAI 12, AISC* 151, pp. 289-296, © Springer-Verlag Berlin Heidelberg.

Caire, G., 2007. *JADE tutorial: JADE programming for beginners*, Telecom Italia Laboratory.

Ferber, J., 1995. Les Systèmes Multi-Agents. Informatique Intelligence Articielle, *The book*, InterEdition.

Gong, L., Mueller, M., Prafullchandra, H., Schemers, R., 1997. Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2, In *Proceedings of the USENIX*, California.

Ko, H., Ramos, C., 2009. A Study on Security Framework for Ambient Intelligence Environment (ISyRAmISF : ISyRAmI Security Framework)", in *Fifth International Conference on Wireless and Mobile Communications*.

Maña, A., López, J., Ortega, J., Pimentel, E., 2004. A Framework for Secure Execution of Software. *International Journal of Information Security*.

Maña, A., Muñoz, A., 2006. Mutual Protection for Multiagent Systems. In: *proceedings of the Third International 3rd International Workshop on Safety and Security in Multiagent Systems* (SASEMAS '06)

Maña, A., Muñoz, A., Serrano, A., 2007. Towards Secure Agent Computing for Ubiquitous Computing and Ambient Intelligence. *UIC* 2007, LNCS 4611, pp. Springer-Verlag Berlin Heidelberg.

Necula, G., 1997. Proof-Carrying Code. In: *Proceedings of 24th Annual Symposium on Principles of Programming Languages*.

Stajano, F., 2010. Security Issues in Ubiquitous Computing, *Handbook of Ambient Intelligence and Smart Environments*, Springer LLC 2010.

Tapia, D., 2010. Agents and ambient intelligence: case studies. *Journal of Ambient, Intel Human Comput.*

TCG, 2005. *Trusted Computing Group: TCG Specifications*. Available online at, https://www.trustedcomputinggroup.org/specs/

Weber, W., Rabaey, JM., Aarts, E., 2005. *Ambient Intelligence*. Springer, New York.

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