Security Analysis of a Multi-Agents System in EU's DEEPSIA Project

Francisco Gomes Milagres
Edson dos Santos Moreira
USP – ICMC
Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação
Trabalhador Sancarlense, 400. 13566-590
São Carlos – SP – Brasil
+55-16-2738161
{milagres, edson}@icmc.usp.br

João Paulo Pimentão
Pedro Alexandre da Costa Sousa
UNINOVA – Centre for Intelligent Robots
Universidade Nova de Lisboa
Quinta da Torre – 1825-114
Caparica – Portugal
+351-21-2948340
{pim, pas}@uninova.pt

ABSTRACT
The objective of DEEPSIA (Dynamic on-linE IntErnet Purchasing System based on Intelligent Agents) is to develop a computational infrastructure that supports companies as purchasers in electronic commerce e–procurement processes, using a multi-agent system, which components may, and effectively are, distributed in four countries (Brazil, Poland, Portugal and Spain). Being a system that uses the Internet and deals with critical information, a high security level is required. The objective of this paper is to present the architecture of the DEEPSIA multi-agents system and to describe the vulnerabilities and the threats such a system faces. The focus is then put on the work being developed by the Brazilian and the Portuguese teams towards the enhancement of the security of such systems.

Categories and Subject Descriptors
K.4.4 [**Computing Milieux**]: Computers and Society – Security, Distributed commercial transactions

General Terms
Design, Security.

Keywords
Mobile Agents, Security, Electronic Commerce, Internet.

1. INTRODUCTION
Since the e–commerce (EC) expression was for the first time defined, new ways were also defined to make business over digital networks. E–commerce represents the act of conducting business communication and transactions over networks and through computers. This term was precisely defined by *The Free On-line Dictionary of Computing* as "the buying and selling of goods and services, and the transfer of funds, through digital communications." It represents a new paradigm that merges the standards, simplicity and connectivity of the Internet with the core processes of companies related to commercialization. [1]

To fulfill this requirement, the DEEPSIA Consortium (supported by *Information Society Technologies Programme* from European Community [1] – IST-1999-20483) aims to address the purchasing business process within Small and Medium Enterprises (SMEs) with a customizable e-commerce application.

One of the greatest challenges of this project is to find out new and better ways to retrieve information from suppliers' web pages. The Internet, commonly seen as the World Wide Web, is considered as a great source of information which can be accessed from any computer. However, the truth is that the Internet is only huge source of data. The process of deriving Information from that data is only achieved by the human brain capacity.

The use of existing autonomous systems in the WWW to find relevant information to be used in electronic commerce, — as can be easily performed by search engines like Google [2], for instance — does not usually produce meaningful results according to the client's specific needs.

The marketplace appears to be an interesting option to solve this problem. It focuses on some business segment, with the most relevant companies of that segment classified together. A purchaser aiming to find some specific vendor information of this segment, using the correct marketplace can easily find it. However, the results are restricted to companies that are members of the marketplace and in many cases, this system is not completely customizable by each marketplace client and does not fit for everyone's necessity.

The direct transaction option (the so called B2B – *business to business*) is turning into one of the most popular business models nowadays. It is an electronic transaction between companies that have already established some formal commercial relation and can easily exchange information directly through the Internet. Similar to the marketplace solution, it is also restricted to a dedicated community and the costs to implement this kind of solution can be prohibitive,

[1] http://www.cordis.lu/ist

[2] http://www.google.com

especially for small and medium sized enterprises, which represent the largest number of companies around the world.

This paper aims to briefly present the architecture of the DEEPSIA's Multi-Agent System and describe its vulnerabilities and the threats it faces, focusing on the Research and Development being undertaken to circumvent these problems.

This paper is divided as follows: section 2 presents the DEEPSIA Project; section 3 describes some concepts about software agents; section 4 discusses security regarding to software agents; section 5 presents the basic security requisites for DEEPSIA's Multi-Agent System; section 6 describes the first approach for security into DEEPSIA, the S-KQML approach; section 7 describes the second approach for DEEPSIA security, "Split and Merge", based on communication security; finally, section 8 makes some end remarks.

2. THE DEEPSIA PROJECT

2.1 Scope

The main objective of DEEPSIA (Dynamic on-linE IntErnet Purchasing System based on Intelligent Agents) is to address the purchasing business process within Small and Medium Enterprises (SMEs) with an e-commerce application, helping to perform usual day-to-day purchasing tasks taking advantage of the potential of the WWW.

The system under development is based on a Multi Intelligent Agent System that autonomously generates an electronic catalogue of products. This catalogue gathers products data from multiple vendors so it can be easily compared. In this way, it is expected to achieve cheaper and more time effective purchases using these search results. [2]

The DEEPSIA's main target is to change the traditional e-commerce business model, which generally considers SMEs mainly as suppliers (e.g., in virtual shops or marketplaces).

The Consortium is composed of the following University partners: UNINOVA (Institute for the Development of New Technologies – New University of Lisbon), ULB (Université Libre de Bruxelles), University of Sunderland; and the commercial partners: ComArch S.A. (Poland), Atlante (Spain), Zeus Consulting S.A. (Greece). The University of São Paulo, from Brazil, is the invited member from outside the European Community.

For further project details, visit its web site at http://www.deepsia.com.

2.2 The Multi-Agents System

The main technologies used by the DEEPSIA project are intelligent agents, ontologies (vocabularies of basic terms and precise specifications of what those terms mean), artificial intelligence, catalogue management, software mediation, database technology and communication technology. The kernel of the DEEPSIA system is the Multi-Agent System (MAS).

The MAS (which general architecture is illustrated in Figure 1) is composed of several agents that follow two basic mechanisms to collect information: the autonomous search (which is performed using a crawler agent) and the direct search (which is performed using a Portal Interface Agent — PIA). The PIA can be easily installed in commercial web sites characterizing the collector agent, which is the multi-agent system interface. The commercial web sites, in this case, form a community of companies that, by realizing DEEPSIA's potential, provide data directly to the system's catalogue. [4]

The autonomous search can be understood in the Figure 1 by following the path that begins with the Crawler agent, goes through the Dispatcher agent, the Miner agent, the Ontology agent and then the Ambiguity removal agent.

The Dispatcher agent is responsible for the catalogue's interaction. It has the ability to choose a Crawler for the catalogue to use and make it crawl with the selected URL.

The Crawler agent is in charge of navigating through the link structure associated with the URL given and classifying the selected pages with the user's themes of interest (in the specific case of DEEPSIA it identifies the pages that are selling products). The process is based on text classification methods, which detailed presentation is out of the scope of this paper. [5, 6]

When the Crawler agent finds interesting pages, it sends these pages to a FTP (File Transfer Protocol) server. Then, the Retriever agent obtains those interesting pages from this FTP server and notifies the Miner agent about new pages found.

The Miner agent has the responsibility of identifying the relevant concepts included in the pages selected by the Crawler. In DEEPSIA, it extracts product information from the pages using heuristic methods.

In that sense, the Miner performs page analysis using heuristic concept identification rules, to identify user's interesting concepts. Those, in the case of DEEPSIA, are related to the available products.

For every concept identified, the Miner agent collects concept information using the predefined rules. Then, the Miner makes use of a knowledge base with the purpose of identifying relevant data. The relevant data is described using concept ontology, enabling the agent with the capability to identify the concepts present in the web pages. In DEEPSIA, in order to be able to collect product information, the knowledge of the Miner will consist of processes to allow retrieving product information from tables, since this is the most common way of presenting product information in a set of analyzed sites.

If the Miner is able to extract interesting content from the page (in this case, product information like its description and price, for instance), it then sends a query to the Ontology agent to determine the class that the product belongs to. [7, 8]

The Miner will then wait for the answer of the Ontology agent. During that period, the Miner will process the rest of the page's products and when it finishes, it will continue to process other pages sent by the Retriever Agent. When the Miner receives the answer regarding the ontology classification, it decides whom to send the product information to. If the product belongs to a single class, the product information (including the class) is directly sent to the assigned catalogue. If the ontology is unable to classify the page or if there is more than one class assigned to the product, the Miner cannot decide upon the correct class for

the product and sends all the information to the Ambiguity removal agent for user assessment.

The Ambiguity removal agent is represented by an interface for human processing. In this case, a domain expert classifies the product, helping the Ontology agent to find out the concept in its ontology base. In this case, the administrator selects the product's concept manually.

The Dynamic Catalogue is the user interface, and it is responsible for presenting the multi-agents' collected information based on users' preferences. The catalogue holds information about the data selected by the agents on the web, the sites contacted, the ontology in-use and all user requirements. The data of the electronic catalogue will be stored in a database and will be made available through a web-browsing interface [3].

It is important to point out that the Miner is supposed to ignore the limitation of the ontology and report all concepts found, either to the catalogue or to the user interface.

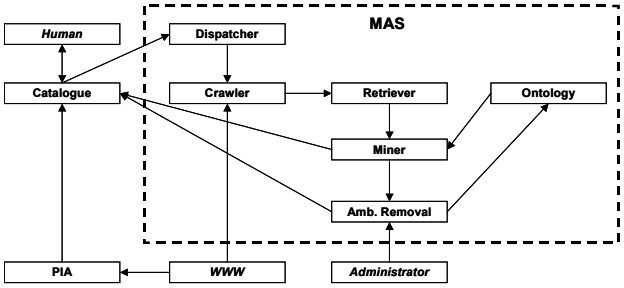

Figure 1. DEEPSIA's simplified architecture. [3]

3. SOFTWARE AGENTS

For the past years, computer systems have evolved from centralized and monolithic supporting centralized applications to networked environments allowing complex forms of distributed computing. A new phase of evolution is under way based on software agents.

A software agent can also be defined as a program that exercises the authority of an individual or organization, working autonomously towards a goal, to meet and interact with other agents. Possible interactions among agents include things such as contract and service negotiation, auctioning, and exchange. [9]

Software agents may be either stationary or mobile. Stationary agents remain resident at a single platform (computer or other "agent enabled" device), while mobile agents are capable of suspending activity on one platform and moving to another, where they resume execution. The concept of mobile code is not new, dating back to the 1960's when remote job entry systems were used to submit programs to a central computer. Recently, code mobility has been popularized through the use of web browsers to download Java applets from web servers. Mobile agents go one step further, allowing the complete mobility of software among supporting platforms to form large-scale, loosely-coupled distributed systems. For further information about software agents in general, refer to a specific agents' research group. [3] [10]

In the DEEPSIA project current status, with a prototype system available through DEEPSIA's web site, the scenario is a set of software agents, running in fixed platforms spread throughout Europe and Brazil, with the agents communicating by KQML messages. [7, 8]

DEEPSIA's architecture has since been analyzed and there is now a common agreement among partners that, considering communication costs and efficiency, some agents should be localized where their work takes place.

One of such examples is the Web Crawler. The major effort on crawling takes place within a specific site and the result of its effort is a set of pages that satisfies a given criterion. Presently, the crawlers are located in Portugal and all the crawling (i.e. fetching the web pages) is done from this location; regardless of the fact that most of the pages fail to meet the criterion. The efficiency would surely improve if the crawling was to be done in the country where the web site is located.

4. SECURITY IN SOFTWARE AGENTS

As the sophistication of mobile software increases, the associated security threats and vulnerabilities also increase. The focus of this paper is on security issues that affect mobile agents, outlining the agents and their mechanism of communication in DEEPSIA's Multi-Agent System and the possible countermeasures for dealing with those issues.

Threats to the security of mobile agents generally fall into four comprehensive classes: disclosure of information, denial of service, corruption of information, and interference or nuisance. [9]

The components of an agent system are going to be used for further delineate threats by identifying the possible source and target of an attack regarding to elements within that paradigm. It is important to note that many of the threats that are discussed have counterparts in classical client-server systems and have always existed in some form in the past (e.g., executing any code from an unknown source either downloaded from a network or supplied on physical media).

Mobile agents simply offer a greater opportunity for abusing and misusing, broadening the scale of threats significantly. New threats arising from the mobile agent paradigm are due to the fact that against the usual situation in computer security where the owner of the application and the operator of the computer system are the same, the agent's owner and system's operator may be different.

There are a number of models for describing agent systems; however, for discussing security issues it is sufficient to use a very simple one, consisting of only two main components: the agent itself and the agent platform. [9, 11]

An agent comprises the code and state information needed to carry out some computation; multiple agents cooperate with one another to carry out some application; mobility allows an agent

[3] http://agents.umbc.edu

to move or hop among agent platforms. The agent platform provides the computational environment in which an agent operates and the platform where an agent originates is referred as the home platform and normally is the most trusted environment for an agent. One or more hosts may comprise an agent platform, and an agent platform may support multiple locations or meeting places where agents can interact. Since some of these aspects do not affect the discussion of security issues, they are omitted from the agent system model used into this discussion, which depicts the movement of an agent among several agent platforms (Figure 2). [12, 13]

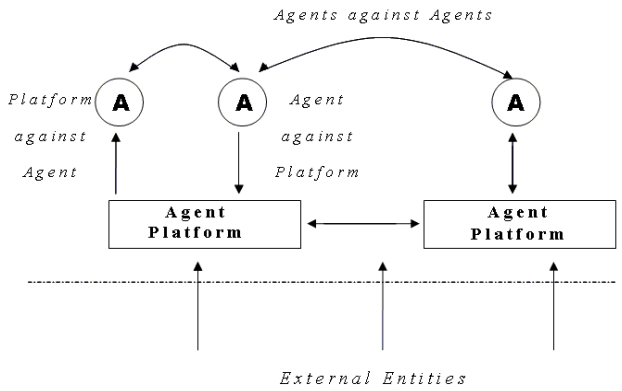

Figure 2. Model of threats in software agents. [11] *apud* [7]

Four threat categories can be identified from the simplified model on Figure 2:

- An agent attacking an agent platform;
- An agent platform attacking an agent;
- An agent attacking another agent;
- External entities attacking the agent system, which is composed by the agents and the agent platforms.

The cases of an agent attacking an agent on another agent platform and of an agent platform attacking another platform are covered within the last category, since these attacks are primarily focused on the communications capabilities of the platform to exploit potential vulnerabilities. The last category also includes more conventional attacks against the underlying operating system of the agent platform. [14]

4.1 Agent against Agent Platform

The mobile agent paradigm requires an agent platform to accept and execute code developed elsewhere. An incoming agent has two main lines of attack: the first is to gain unauthorized access to information residing at the agent platform and the second is to use its authorized access in an unexpected and disruptive fashion. Unauthorized access may occur simply through a lack of adequate access control mechanisms at the platform or weak identification and authentication, which allows an agent to masquerade as another agent, trusted by the platform.

Once access is gained, information residing at the platform can be disclosed or modified. Besides confidential data, this information could include the application code of the platform itself. Depending on the level of access, the agent may be able to either alter the code or shut down the platform. Even without gaining unauthorized access to resources, an agent can deny platform services to other agents by exhausting computational resources, if resource constraints are not established or not tightly set out. Otherwise, the agent can merely interfere with the platform by issuing meaningless service requests wherever it is possible, causing denial of service attacks.

4.2 Agent Platform against Agent

A receiving agent platform can easily isolate and capture an agent and may attack it by extracting information, corrupting or modifying its code or state, denying requested services, or simply reinitializing or completely terminating it. Extracting electronic cash directly from the agent is one simple example. An agent is very sensitive to the interaction with the agent platform and may be corrupted merely by having the platform responding falsely to requests for information or service, altering external communications or delaying the agent until its task is no longer relevant.

Extreme measures include the complete analysis and reversing engineering of the agent's design so that subtle changes can be introduced. Modification of the agent by the platform is a particularly insidious form of attack, since it can radically change the agent's behavior (e.g., turning a trusted agent into malicious one) or the accuracy of the computation (e.g., changing collected information to yield incorrect results).

4.3 Agent against other Agents

An agent can target another agent using several general approaches. These include actions to falsify transactions, eavesdrop upon conversations or interfere with an agent's activity. For instance, a malicious agent can respond incorrectly to direct requests it receives from a target or deny that a legitimate transaction occurred. An agent can gain information by serving as an intermediary to the target agent (e.g., through a masquerade attack — type of attack in which one system entity illegitimately assumes the identity of another entity) or by using platform services to eavesdrop on intra-platform messages. [15]

If the agent platform has weak or no control mechanisms, an agent can access and modify data or code of another agent or interfere with the agent by invoking its public methods (e.g., attempt buffer overflow or reset to initial state). Even with reasonable control mechanisms in place, an agent can send messages repeatedly to other agents in an attempt to deny them the ability to communicate.

4.4 External Entities against Agent System

Even assuming that the locally active agents and the agent platform are well behaved, other entities (both outside and inside the agent system) may attempt actions to disrupt, harm, or subvert the agent system. The obvious methods involve attacking the inter-agent and inter-platform communications through masquerade, (e.g., through forgery or replay attacks - an attack in which a valid data transmission is maliciously or fraudulently repeated, either by the originator or by an adversary who intercepts the data and retransmits it) or intercept. [16]

For example, at a level of protocol below the agent-to-agent or platform-to-platform protocol, an entity may eavesdrop on

messages in transit — in DEEPSIA specific case, in KQML — to and from a target agent or agent platform to gain sensitive information. An attacking entity may also intercept agents or KQML messages in transit and modify their contents, or simply replay the transmission dialogue at a later time in an attempt to disrupt the synchronization or integrity of the agent system. Denial of service attack through available network interfaces is another possibility.

5. BASIC SECURITY REQUIREMENTS

The basic security requirements that are going to be presented here for KQML — the Agent Communication Language (ACL) in use by the DEEPSIA's Multi-Agents System at the present time — are based on the analysis of the security models for Privacy Enhanced Mail (PEM), Common Object Request Broker Architecture (CORBA), Distributed Computing Environment (DCE) and Secure Infrastructure for ACLs (S-KQML) — a previously defined secure infrastructure for agent communication languages based on the original KQML. [17, 18, 19, 20]

The security capabilities that should be supported by this model include: [20, 21]

- *Authentication of principals*: Agents should be capable of proving their identities to other agents and verifying the identity of other agents;
- *Preservation of message integrity*: Agents should be able to detect intentional or accidental corruption of messages;
- *Protection of privacy*: The security architecture should provide facilities for agents to exchange confidential data;
- *Detection of message duplication or replay*: A rogue agent may record a legitimate conversation and later play it back to disguise its identity. Agents should be able to detect and prevent such playback security attacks;
- *Non-repudiation of messages*: An agent should be accountable for the messages that they have sent or received (i.e., they should not be able to deny having sent or received a message);
- *Prevention of message hijacking*: A malicious agent should not be able to extract the authentication information from an authenticated message and use it to masquerade as a legitimate agent.

The architecture here proposed is a basic security model, which supports authentication of sender message integrity and privacy of data. An enhanced security model should additionally provide non-repudiation of origin, proof of sending and protection from message replay attacks, including support to frequent change of encryption keys to protect from cipher attacks. [20, 22, 23]

6. FOCUSING ON ACL SECURITY: THE S-KQML APPROACH

In order to implement this security architecture, the S-KQML model proposes several new KQML parameters and some modifications to a proposed standard ontology for agents. [20, 22]

It is assumed that KQML-speaking agents use a basic agent ontology which provides a small set of classes, attributes and relations helpful in talking about agents, their properties and the relationships and events in which they partake.

Assuming this ontology, the S-KQML architecture introduces a new sub-class of agent named *authenticator* and a new relation *key/5* which describes a key used by an agent, as shown below: [20]

```
(key            <sending-agent>
                <receiving-agent>
                <master-key?>
                <key-type>
                <encrypted-key>)
```

An instance of this relation specifies a key that the sending agent will use in secure communication with the receiving agent. If the third argument is true then the key is a master key, else it is a session key. If the receiving agent is a variable, then the key is used by the sending agent to communicate with all other agents. Note that this would typically be the case for asymmetric keys. We assume that agent addresses are represented in this ontology with the *address/3* relation. [23, 24]

```
(address        <agent>
                <transport>
                <transport-address>)
```

Instances of this relation specify transport addresses for the agent given in the first argument, as shown below:

```
(address        agent1
                smtp
                10.0.0.1)
(address        agent2
                tcpip
                (10.0.0.1 8080))
```

These addresses are known as special agents, such as *agent name servers* and *authenticator agents*.

6.1 New KQML parameters

Several new KQML parameters are required to implement the proposed security architecture: [20]

```
:auth-digest   (<digest-type>   <encrypted-digest>)
```

The *digest-type* specifies the hashing function used (e.g., MD5) to compute the message's digest. The *encrypted-digest* is the message's digest encrypted using the key specified by the *:auth-key parameter*. This parameter should be present to prevent message hijack and to provide sender authentication and integrity assurance. [23]

```
:auth-key   (<bool>   <key-type>   <encrypted-key>)
```

This parameter specifies the key being used to encrypt any *:auth-digest* parameters present. If the first element of the triple is true then the master key is used, otherwise, the session key is used.

6.2 New KQML performatives

The following new KQML performatives were added to standard KQML in order to allow the implementation of this architecture:

`auth-link`

The message sender wishes to authenticate itself to the receiver and set up a session key and message ID for a secure connection using this performative.

`auth-challenge`

The sender challenges the identity of the receiver in response to an *auth-link*. The sender then encrypts a random string using the master key $K_{s,r}$ or K_s and sends it as a *:content*.

`auth-private`

When the sender is posting a confidential message to the receiver, the content parameter contains the encrypted message and the *auth-key* parameter specifies the encryption key. The *:auth-digest* parameter should be present to verify the identity of the sender and the *:auth-msg-id* and *:auth-key* parameters may be present if an enhanced security model is required.

6.3 The Secure KQML Model

An implementation of S-KQML should support the following protocol to conform to the basic security model. This model basically supports authentication, integrity and privacy of data in transit. If asymmetric keys are used for session and master keys, this model also supports non-repudiation of origin. [20]

When *Agent_A* sends a secure message to *Agent_B*, it would compute a message digest and encrypt it using the master key (as indicated by the value K for the *:auth-key* parameter). [22]

```
<performative>
        :sender Agent_A
        :receiver Agent_B
        :auth-key K
        :auth-digest (<digest-type>
                  <encrypted-digest>)
        ...
```

Alternatively, if *Agent_A* needs to send a confidential message to *Agent_B*, it can encrypt the message and embed it in an *auth-private* performative, like shown below:

```
auth-private
        :sender Agent_A
        :receiver Agent_B
        :auth-key K
        :auth-digest  (<digest-type>
                  <encrypted-digest>)
        :content <encrypted-KQML-msg>
```

This proposed model can be used when the *:sender* does not know the *:receiver* in advance, e.g., for messages to be broadcasted, routed by some other specific agent or if *Agent_A* and *Agent_B* do not require prevention of message replay and can afford the cost of using the master key during all the communication session, for instance.

In the previous message, the *:auth-digest* parameter can be used to verify the integrity of the message, authenticate the sender and ensure non-repudiation of origin (if the master key is asymmetric). If the message has been corrupted, the message digest will not agree with the value of the *:auth-digest* parameter. Since the message digest is encrypted with the master key of *Agent_A*, only itself or the agents with which the sender shares the encryption key could have generated the message.

If the master key is an asymmetric key, only the message sender could have generated the message, as only the sender knows the private key that has been used for encryption. Note that this method can only verify the identity of the generator (i.e., if the message was encrypted by the sender agent of the message). This message can be a replay of a legitimate message previously sent by the generator.

6.4 Limitations of the proposed Secure KQML model

The secure model that is proposed in this paper for KQML has a number of limitations which now are going to be briefly enumerated.

- *Credentials*: This model does not provide a mechanism to exchange credentials, that is, for one agent to empower another to act on its behalf;
- *Non-repudiation of receipt*: This model does not support non-repudiation of message receipt. This can be a very useful capability, but would be difficult to implement due to the asynchronous nature of KQML and can be done only at the application level;
- *Messages to unknown receivers*: Although one enhanced security model could support message replay detection, the proper use of the *:auth-msg-id* parameter is required. This requires that the recipient is known in advance. One of the essential features of KQML is the use of facilitator class agents (e.g., brokers and proxy agents to automatically rout messages whose intended recipients are described in general terms by the sending agent);
- *Stateless*: The security architecture requires that agents maintain state information. The agents can choose not to use this feature if they are not concerned that message replay attack and cipher attack;
- *Crypto-awareness*: An agent can send out authenticated messages if and only if it has crypto capabilities;
- *Constraints on delivery*: Messages delivery must be reliable and in order. (A fair limitation considering that KQML itself assumes that);
- *Use of recommended APIs*: The model should be enhanced to support the use of the Crypto APIs recommended by NSA (National Security Agency), especially for the *key-type* and *digest-type* values, due to cryptography export international regulations.

Some of these limitations stem from the basic features of KQML; others, according to other secure KQML implementations and ACLs researches can be lived with and the rest could be addressed by if required by updating the KQML

architecture or substituting this ACL by FIPA (Foundation for Intelligent Physical Agents) ACL, for example, which is a default language for agents' communication nowadays. For more details about FIPA ACL and its development, visit its web site at http://www.fipa.org.

The proposed KQML security model addresses privacy, authentication and non-repudiation (if asymmetric key mechanism is used for the master and session keys) in agent communication. It does not fully address the issue of message replay, especially if the recipient of a KQML performative is not known in advance. Ultimately, this security model depends on definitive changes on default ACL by the researches community, which tends to be FIPA, instead of DEEPSIA's KQML suggested implementation.

According to Tim Finin, one of the KQML creators, "For the most part, the FIPA standards have supplanted KQML". [7, 8]

7. FOCUSING ON COMMUNICATION SECURITY: THE SPLIT AND MERGE APPROACH

One of the issues of research focus is the way of securing communication within the system.

It has been shown that the traditional approaches to message encryption (who rely strongly on the lack of computational power to perform complex mathematical tasks) are being undermined by the increasing of microprocessor speed (Figure 3).

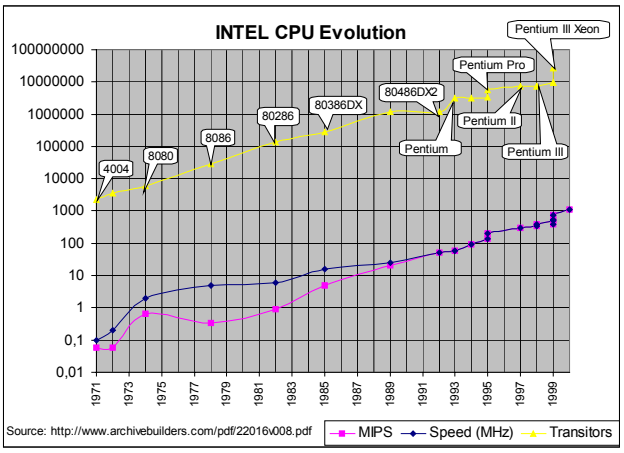

Figure 3. Evolution of INTEL CPUs

It is our belief that with the promised increase of computational capacity (e.g., quantum computing) the solution should not rely on mathematically complex algorithms.

To this end, UNINOVA has been developing a method ("Split and Merge") as an alternative to the methods currently in use. [25]

In the Split and Merge algorithm, the security of the message does not rely on the ability of being able to decipher the message, but on the ability of getting the message itself. It does not deal with obscuring the contents of the message, but in splitting the message in parts (as many and as small as whished) so that the possession of a part does not give information about the content of the message. The next key concept has to do with routing different pieces of the message through different paths, while going from the source to the destination. The routing of each piece is done randomly so that the paths followed by each piece are probably different. The purpose of routing the pieces through different paths is to decrease the possibility of a perpetrator compromising all possible nodes on the route of the message. Further details of this algorithm can be found on [25].

The basis for the security of the algorithm is on the small probability an attacker has of being able to secure enough nodes to guarantee catching enough parts of the message to enable its understanding.

The work currently under way is on the definition of a mathematical model for the system, in order to, given a network of nodes, determine which and how many nodes must the attacker control to have access to a given percentage of a message sent by a given source node.

Once this model is defined, it will be possible to measure, for each different network, the gain of the system in terms of the effort needed to decipher the message. This effort should then be added (if cipher is used) to the effort needed to break the ciphering mechanism used.

It has been noticed that if a Cipher Block Chaining (CBC) method is used and the attacker does not hold the first block of the message, the message is undecipherable even in possession of the key. This may primarily be used for improving efficiency of the algorithm by using Split and Merge only for the first block and sending the rest of the message directly to the destination.

8. CONCLUSIONS AND FINAL REMARKS

This paper presents the status of the European Commission's IST DEEPSIA Project. It describes the system being developed as a tool to assist in e-procurement. It briefly describes the architecture of the system (that uses a multi-agents system), going into some detail as the components of the system, their functionalities and interactions are described. The focus is then on the underlying technology (software agents) and on the security threats such systems suffer.

The paper concludes with a description of the efforts currently under way by the Brazilian and Portuguese project teams with regards to, respectively: a security enhancement of the KQML language currently being used and a specification of a new method to deal with protection of privacy of the messages being exchanged.

9. ACKNOWLEDGMENTS

We would like to express our gratitude to all members of DEEPSIA Consortium (through IST-1999-20483). Our thanks also go to the Brazilian funding agency CNPq (Process n° 680263/01-2), that supports the Brazilian research group. We also thank the anonymous referees for their useful comments.

10. REFERENCES

[1] Howe, D. "E-Commerce" *The Free On-line Dictionary of Computing*, 1993-2002. http://www.dictionary.com

[2] DEEPSIA Consortium. *Technical DEEPSIA Annex 1: Description of Work*. July, 2000. 87 p. Report. IST PROJECT-1999-20483.

[3] Garção, A.S.; Sousa, P.A.; Pimentão, J.P.; Santos, B.R.; Blazquéz, V.; Obratanski, L. *Annex to DEEPSIA's Deliverable 4 — System Architecture*. January, 2001. 135p. Report. IST PROJECT-1999-20483.

[4] "What is Ontology?" *Ontology.org Frequently Asked Questions*. 2000. http://www.ontology.org/main/papers/faq.html

[5] Sousa, P., Pimentão, J., Garção, A., "DEEPSIA - From supply chains to supply webs". In *Intelligent Engineering Systems through artificial neural networks*, Cihan H. Dagli, Anna L.Buczak, Joydeep Ghosh, Mark J. Embrechts, Okan Ersoy, StephenHercel, Volume 11, ASME PRESS, NEW YORK, ISBN 0-7918-0176-4, 2001, pp. 1019-1024.

[6] Sousa, P., Pimentão, J., Garção, A., "DEEPSIA – Focusing e-commerce on the purchaser's side", In *International ICSC Congress on Computational Intelligence: Methods and Applications* (CIMA'2001), Ludmila l. Kuncheva, Friedrich Steimann, Christian Haefke, Mayer Aladjem, Vilem Novak, ICSC Academic Press, Canada, ISBN 3-906454-26-6, 2001, pp. 436-442.

[7] Finin, T.; Weber, J. *Specification of the KQML Agent-Communication Language*. The DARPA Knowledge Sharing Initiative, 1993.

[8] Finin, T.; Labrou, Y. *A Proposal for a new KQML Specification*. University of Maryland Baltimore Count (UMBC). Baltimore, 1997.

[9] Jansen, W.; Karygiannis, T. *Mobile Agent Security. Technical Report*, National Institute of Standards of Technology (NIST). Baltimore, 1999.

[10] Fuggetta, A.; Picco, G. P.; Vigna, G. Understanding Code Mobility. *In IEEE Transactions on Software Engineering*, 24(5), May 1998, pp. 342-361.

[11] Uto, N.; Dahab, R. Segurança de Sistemas de Agentes Móveis. In *III Simpósio de Segurança em Informática*, São José dos Campos, SP, Brazil. 2001.

[12] FIPA *Agent Management Specification. Foundation for Intelligent Physical Agents*, 2000. http://www.fipa.org/specs/fipa00023/

[13] White, J.E. *Mobile Agents. In: Software Agents*, J.M. Bradshaw (Ed.), Menlo Park, Calif., AAAI Press, 1997.

[14] Jansen, W. *Countermeasures for Mobile Agent Security. In Computer Communications*, Special Issue on Advances in Research and Application of Network Security, November 2000.

[15] "masquerade attack", *The LinuxSecurity.com Dictionary*. http://www.linuxsecurity.com/dictionary/dict-249.html

[16] "replay attack", *The LinuxSecurity.com Dictionary*. http://www.linuxsecurity.com/dictionary/dict-330.html

[17] RFC 1421: *Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication* Procedures, 1993. http://www.ietf.org/rfc/rfc1421.txt

[18] *Common Object Request Broker Architecture Security Suite*, 2000. http://www.omg.org/technology/documents/formal/omg_security.htm

[19] *Security in the Distributed Computing Environment. Security on the Web Using DCE Technology*, Document Number SG24-4949-00. IBM Corporation, 1997. http://www-3.ibm.com/software/network/dce/library/redbooks/sg244949/4949c112.htm

[20] Thirunavukkarasu, C.; Finin, T.; Mayfield, J. Secret Agents – A Security Architecture for the KQML Agent Communication Language. In *Intelligent Information Agents Workshop held in conjuntion with Fourth International Conference on Information and Knowledge* Management. Baltimore, 1995.

[21] Spafford, G.; Garfinkel, S. *Practical UNIX & Internet Security*. O'Reilly & Associates. 2 nd ed. April, 1996.

[22] Milagres, F. G.; Moreira, E. S. "*Especificação de Segurança na Comunicação de Agentes*" ("*Specification of Security on Agent Communication Languages*"), In Módulo Security News number 251 and on-line at Módulo Security Solutions web site, Academic Research category. Módulo Security Solutions S. A. São Paulo. July 2002. (in portuguese) http://www.modulo.com.br/pdf/milagres-deepsia_security.pdf.

[23] Schneier, B. *Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., 2nd ed. January, 1996.

[24] Diffie, W.; Hellman, M. E. New directions in cryptography In *IEEE Transactions on Information Theory 22* (1976), 644-654.

[25] Pimentão J, Sousa P, Garção A, "*Split and Merge - an algorithm to implement security on the Internet*". In Communications World, N. Mastorakis ed., WSES Press, 2001.