KASS: The KQML Agent Security System

Bilel ELAYEB Mohamed BEN AHMED
RIADI-GDL laboratory, The National School of Computer Sciences (ENSI)
Manouba University, 2010 Manouba, Tunisia.
{Bilel.Elayeb, Mohamed.Benahmed}@riadi.rnu.tn

Abstract

Multi-agent systems communicating using KQML gain more and more popularity and importance. With this evolution, these systems are increasingly likely to be the target of various disordered states against their security, such as congestions, malevolent accesses and attacks. Thus, it becomes inescapable to provide these systems with tools and mechanisms able to inhibit these disordered states.

One of anti-attacks measurements we focused on within the framework of this work is the KQML Agent Security System (KASS). In fact, our contribution is based, on the one hand, on an extension of the KQML message describing how the content of the message was made safe, and on the other hand, on the security module and its units which try to resolve the limits of the current systems architectures. Indeed, thanks to our new approach, partial encryption of the message becomes possible and independent of the platform. Security becomes independent of the agent and takes account of the confidential data and of the inaccessibility of the central authority, too.

1. Introduction

Security in multi-agent systems can be implemented either by the *message encryption* (security against monitoring by undesirable side) or *message signing* (assuring of message content's integrity). In some cases it is not necessary to secure the whole message but only its parts.

Ther'is a great number of systems and principles allowing secured communication in multi-agent systems (MAS). Using the existing security systems in MAS brings a couple of disadvantages (see Section 2). The proposed approach attempts to avoid them and suggests a set of recommendations how to implement agent's interaction security regardless of a programming language and a multi-agent platform.

A wide variety of potential attacks can be categorized and formalized into general threat model. Under this term, we should understand a general breakdown into passive attacks, in which communications are merely monitored, and active attacks, in which communications or the underlying agents themselves are subverted via deletions, modifications or additions of data to the communication. The most obvious passive attack is simple monitoring of network packets, often accomplished by malicious agent, eavesdropper, using a packet sniffer, which records for later analysis all the packets being transmitted via network among arbitrary number of agents. Even if the communication is encrypted, still there is a risk of traffic analysis, where the eavesdropper monitors the information exchange and agents' relationship. Special attention should be paid to more dangerous active attacks, involving the disruption of the communication paths between agents or the attack of the underlying infrastructure, such as spoofing attack or replay attack, in which malicious agent impersonates the legal one and/or repeats the communication, and modification of messages, corruption of integrity and confidentiality or brute denial of service attacks.

The MAS security architecture design considerations have been motivated by various mentioned security risks. Necessary security services should be defined in order to provide entities within a MAS required support to assure confidence in the distributed system and to prevent it from the identified security attacks. These security instruments should increase trust and confidentiality within and among agent communities/technology and provide security mechanisms, such as encryption, authentication, message integrity services, etc., employing cryptography algorithms. The security mechanisms can utilize existing agent platform mediators such as the Agent Management System (AMS), Directory Facilitator (DF), or Agent Communication Channel (ACC), e.g. for authentication purposes [2].

Proposed Architectures offer on different level the security and trust services that defend MAS against corrupted naming (mapping or matchmaking) services, insecure communication channels, insecure agents delegations, lack of accountability, etc. [3,4]. Essential security services presented in this paper constitute an important part of the overall architecture being developed.

Our approach provides principles for secure messages transfer and defines a KQML message extension for a new element that describes a form of the message security. The KASS-Security solution is open to address inaccessibility of the central authority, which issues security certificates to particular agents (see Section 3.7).

2. Security system requirements

Security mechanisms, described in the published KQML security specification [5] [9] [10], do not allow security of just parts of the message. This property breaches the requirements for openness and implements security for agents residing on an agent platform only and not for standalone agents. In the following we sumup the security requirements and dissimilarities from the FIPA specification:

2.1 Partial message encryption

Possibility to secure not only the whole KQML Message but its parts is required, e.g. possibility to send also a delegation (passed from one agent to another), sign certain part of text or data for storing it in database along with its signature and guarantee its authentication for later use (record of transactions), or encrypt only password required for access to particular resource to allow subsequent detection of a kind of requested data. This can be reached using the structure (class) containing not only carried text (signed or encrypted) but also additional information concerning security (type of security action, created signature, identification of used key). On the receiving side, this structure (class) is processed and the original text obtained.

2.2 Loose link with platform

Possibility not to bind security support tightly to the agent platform. KQML-interoperable agents can run on different platforms (without the implemented type of security) or even can be standalone with no platform. For example, agent collects data from appropriate company database server, running on different operating system, and provides data to MAS (in secure way). It is possible to ensure this including the security directly into the agent (its communication wrapper) or by library supplied with agent.

2.3 Agents' security independence

Avoid agent's core necessity to choose, set type or negotiate about algorithms used in secure communication. These actions have to be done by security module automatically. Negotiation about security algorithms can be time-consuming on occasional connection. If agent sends over such connection message with security algorithm which recipient does not understand, recipient cannot inform the sender about it immediately. Every agent (its security module) has to register (with certain authority) some lists of security algorithms (and public keys) [6], which it uses. Agent that wants to send a secured message has to ask for a list of receiver-supported algorithms and use one of them for secure communication.

2.4 Security data confidentiality

All private keys and other security related data have to be available to their owner only. Data may not be accessible to anyone else (even the agent platform). Platform can be distributed across many computers and hence it is impossible to ensure security within the whole platform, if the private data are managed by platform. Every agent has to keep its private data secured, even during its migration on other platform.

2.5 Central Authority Inaccessibility

The security infrastructure shall enable agents operate in the cases when the central authority becomes inaccessible, overloaded or fails functioning. Even though a central authority is a vital component of the community, various 'modes' of operation reflecting the current functionality of the central authority need to be allowed. Running multiple central authorities will be also possible.

3. KASS-Security Prototype

In the proposed approach the dedicated central authority is required to administer an important part of the security mechanisms. This authority issues appropriate licenses certificates. Agents use issued certificates to prove their identities and execute security related actions within the system [7]. Function of the central authority is in the proposed system implemented by the Security Certification Authority (SCA).

3.1 Certificates and Their Importance

Certificate contains mandatory information requested by SCA and may contain additional information supplied by an agent. Information requested by SCA is agent's identification, public keys (and their description) and requested validity time and security level in MAS. SCA verifies these data and stores them into the certificate. SCA can't guarantee validity of optional data, but can assure their constancy (originality) when providing other agents with the certificate. SCA signs the whole certificate and thus allows receiver to verify the integrity of contained data.

Security level is set up by SCA according to username and password, which agent sent in its registration request.

If agent needs to send encrypted message to another agent, verify signature of received message or check the security level, it asks SCA for particular certificate. Here is an example of certificate issued for the 'agent' agent:

```
certificate-ident SCA_CERTIFICATE_1
sca-ident
    (agent-identifier:name sca@platform.net)
agent-ident
    (agent-identifier:name agent@platform.net)
time-from     Thu Jan 01 00:00:00 ENSI 2004
time-to       Fri Dec 31 23:59:59 ENSI 2004
security-level       VISITOR

key-description
  ident              SIGN_1
  time-from   Thu Jan 01 00:00:00 ENSI 2004
  time-to     Fri Dec 31 23:59:59 ENSI 2004
  type        public-key
  key-param   SHAwithDSA/1024
  key-value   65A7ED89C2………6AC54DF423

key-description
  ident       CRYPT_1
  time-from   Thu Jan 01 00:00:00 ENSI 2004
  time-to     Fri Dec 31 23:59:59 ENSI 2004
  type        public-key
  key-param   RSA/1024
  key-value   6A248DC86B………85293B67DC
```

3.2 Integration of Security into Message

In the proposed system, the agents communicate using KQML Messages according to the KQML standard. A message is extended to contain a new slot called KASS-Security. This slot specifies how the message content has been secured. Extended message may look as follows:

```
 (inform
   :sender          (sender@platform.net)
   :receiver        (receiver@platform.net)
   :language        (LPROLOG)
   :content         („Text to be signed“)
   :KASS-Security  (:type SIGN
          :signature 45A7………30AD
          :certificate-ident SCA_CERTIFICATE_1
          :key-ident SIGN_1 ) )
```

Items of the KASS-Security slot inform that message content was signed (signature is included) and it can be verified by public key SIGN_1 stored in certificate SCA_CERTIFICATE_1.
Next example presents encrypted message:

```
(inform
  :sender           (sender@platform.net)
  :receiver         (receiver@platform.net)
  :language         (LPROLOG)
  :content          („34AD………6BA3“)
  :KASS-Security   (:type CRYPT
          :certificate-ident SCA_CERTIFICATE_1
          :key-ident CRYPT_1 ) )
```

KASS-Security slot items now inform that message content is encrypted by public key CRYPT_1 stored in certificate SCA_CERTIFICATE_1.
Similarly, it is possible to secure only parts of the content. To allow this, it is necessary to create new class/structure containing both, part of the content to be secured and description of its security:

```
class / struct SecurityText
{ String text;      // signed or encrypted text
  String security; // security information
}
```

Following that an example of message containing secured part of the content. Agent asks for registering its computational results and confirms originality of the result with the signature:

```
(inform
   :sender (sender@platform.net)
   :receiver (receiver@platform.net)
   :language (LPROLOG)
   :content
      (action
         (agent-identifier :name … :address…)
    (data-register
            :data (security-text
               :text „10,20,30,40,50“
               :security (security-description
                  :type SIGN
                  :signature 25A8………D6C7
                 :certificate-ident
                 SCA_CERTIFICATE_1
                  :key-ident SIGN_1 ) ) ) ) )
```

In the following example agent asks for data protected by password. This password is transferred as encrypted text.

```
(inform
  :sender   (sender@platform.net)
  :receiver (receiver@platform.net)
  :language (LPROLOG)
  :content
    (action
       (agent-identifier :name … :address … )
       (data-request
         :data-type „type_of_requested_data“
         :password (security-text
          :text „67C5………DA8B“
         :security (security-description
          :type CRYPT
          :certificate-ident SCA_CERTIFICATE_1
          :key-ident CRYPT_1 ) ) ) ) )
```

3.3 Description of SCA's activity

Common security system fails when SCA (or similar central authority) is inaccessible. The system described here also uses SCA and certificates but in a different way. Registered certificates are not stored only in the SCA but after signing they are also sent back to the registering agents. If one agent requires certificate of another one, it should at first ask SCA for it. In cases of SCA inaccessibility, the agent is allowed to ask for it directly from the target agent. Certificate is signed by SCA and therefore its validity can be verified. Now the security can work, even if the SCA is (temporarily) inaccessible.

The described approach also allows the use of security in the area of mobile agents. When two agents meet, they can exchange their certificates and prove their identities. Certificates contain full identification of their owners and are completed with SCA's signature. Using the public keys from the certificates allows the verification that the particular agents own appropriate private keys. Thus, when mobile agent registers its certificate with SCA, it can migrate and still use the certificate to prove its identity.

3.4 Session Keys and Their Use

In case of encrypting a huge amount of data or of frequent communication between two agents, the usage of asymmetric keys is not apposite because it requires considerable computational resources for encryption and decryption. Instead of asymmetric keys, the symmetric session keys can be used. When this situation occurs, security module generates session key and sends it directly (encrypted by asymmetric key) to the other agent. Transferred data are encrypted using the new symmetric temporary key now, as the symmetric key encryption algorithms are not so time/resources consuming. As soon as the communication is finished, the session key is invalidated. Activities related with generating and using session keys are completely assured by the security module.

3.5 Common Agent Key Replacement

After a certain period of time or when the suspicion on the key misusage happens, an agent is allowed to generate new keys. Immediately after generating them, agent asks SCA for new certificate registration and at this moment the previous one becomes invalid. By this way it is possible to register new certificate before the validity time of the old one expires. Every certificate contains unique identification (ID). If an agent signs message using the new certificate, the KASS-Security slot will contain ID of this certificate. Receiver agent does not have a new version of the sender's certificate and has to ask SCA for it. Similar situation occurs when the agent receives message encrypted by invalidated key. In that case, receiver informs sender that used key has been invalidated and for the future, communication requires messages encrypted by the new one. Security module can be set up by agent's core to accept messages encrypted only according to the latest certificate or (for a certain period of time) accept messages encrypted according to older (but still time valid) certificates.

This approach can be used not only for key replacement but also for immediate decreasing of cooperator's security level. This can happen when an agent with a high security level asks SCA to do it.

3.6 SCA Key Replacement

Key replacement of SCA is much complicated than replacing keys of common agent. As the first step, SCA generates new keys and creates new corresponding certificate. In the second step, SCA sends this certificate signed by last valid publicly known key to all registered agents. All of them have to accept the change of the SCA's keys. When SCA has received the confirmations from the registered agents (except inaccessible ones), it sends them their original certificate signed by the new key. Now SCA can start to use the new certificate. Common agent's security module clears its certificate database and this causes requesting for all new necessary certificates from SCA. Other problems are caused by inaccessibility of some agent. From this reason, ID's certificates are changed during the replacing of SCA's certificates too. Thus, mobile agents also can detect this change and ask for their new certificates from SCA as soon as possible. It is only up to SCA how long time after changing its certificates, SCA allows agents to update their certificates.

3.7 SCA Inaccessibility

As was already stated, each agent has its own certificate and these certificates are registered with SCA. During the temporary SCA inaccessibility, the agents are allowed to provide their certificates one other. Even though the agents can't register the new certificates anymore, the already existing security links are not affected. It is true that permanent loss of SCA causes troubles for the communication security. This problem can be solved either by recovering SCA from backup or by setting multiple SCAs.

There could be another SCA operating in the agents' community. This SCA may act as a backup and keeps synchronizing the database with the first one. In other cases, certain more SCAs are required. When the main SCA is lost, the first backup becomes the main one and new backup SCA is created to complete the number of backups. Alternatively, there is no predefined structure in the community of SCAs and the agents can register with any accessible SCA.

In a special situation of the MAS malfunction, no SCA and only some of the common agents could stay active. Then there is no backup of SCA that could be used for its recovery. In such a case, the agents must be able to create new SCA by themselves. First, the agents create a new instance of the empty SCA and give it (during the start-up) the last known certificate of the original SCA (each of active agents has to know it).

New SCA cannot use it for new certificate registration because of not having the private parts of keys but can use the public key from it to verify signatures of other agents' certificates. New SCA generates new keys and certificate for itself. The agents send their certificates confirmed by the old SCA to the new SCA. SCA sends them its new certificates and SCA's certificate signed by its new keys. These certificates are sent only to agents, which certificates new SCA verified by the key of the old SCA. The new certificates are sent encrypted because common agents do not know the public key of the new SCA for signature verification, but SCA has the certificates of these agents.

This way can be also used for creating new SCA by a group of mobile agents, for example to create other secured temporary agents. Mobile agents have to keep their older certificates that are relevant for their home platform.

4. KASS architecture

SCA is a standalone agent that does not affect agents' interaction. However, it needs to start first. See Figure 1 for the placement of SCA in the community.
Security service is provided by security module that is placed between the agent's core and communication layer, as could be seen on Figure 2.

Messages are withdrawn from the input queue. These messages could serve for security management (e.g. required certificates); they could be secured (e.g. by encryption, signature, etc.), or unsecured-that is passed directly to the agent's core.

The queue of outgoing messages contains messages created by security module (e.g. request for certificate) and ones created by the agent's core. The latter are secured according to the requirements of the core. Agent's core is allowed to restrictedly influence behavior of the security module. Inner structure of security module is shown on Figure 3.

Security module contains several individual units:

1. Cryptology unit: it provides the encryption, the decryption, the creation and the check of the message signature.

2. Exchange unit: it provides connection to SCA and exchanges certificates with other agents.

3. Storage unit: it maintains database of received certificates, private keys and session keys. This unit provides them to other units. It is also required to store data safely during agent's migration.

4. Interface unit: it provides interface between security module and agent's core, which is necessary, for instance, to configure the security or when partial encryption is required.

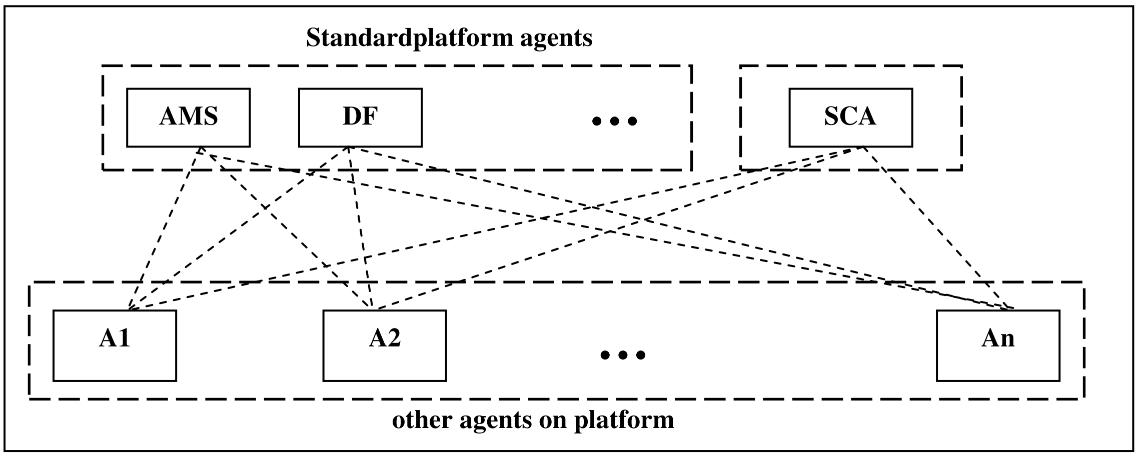


Figure 1 : Agent platform with Security Certification Authority

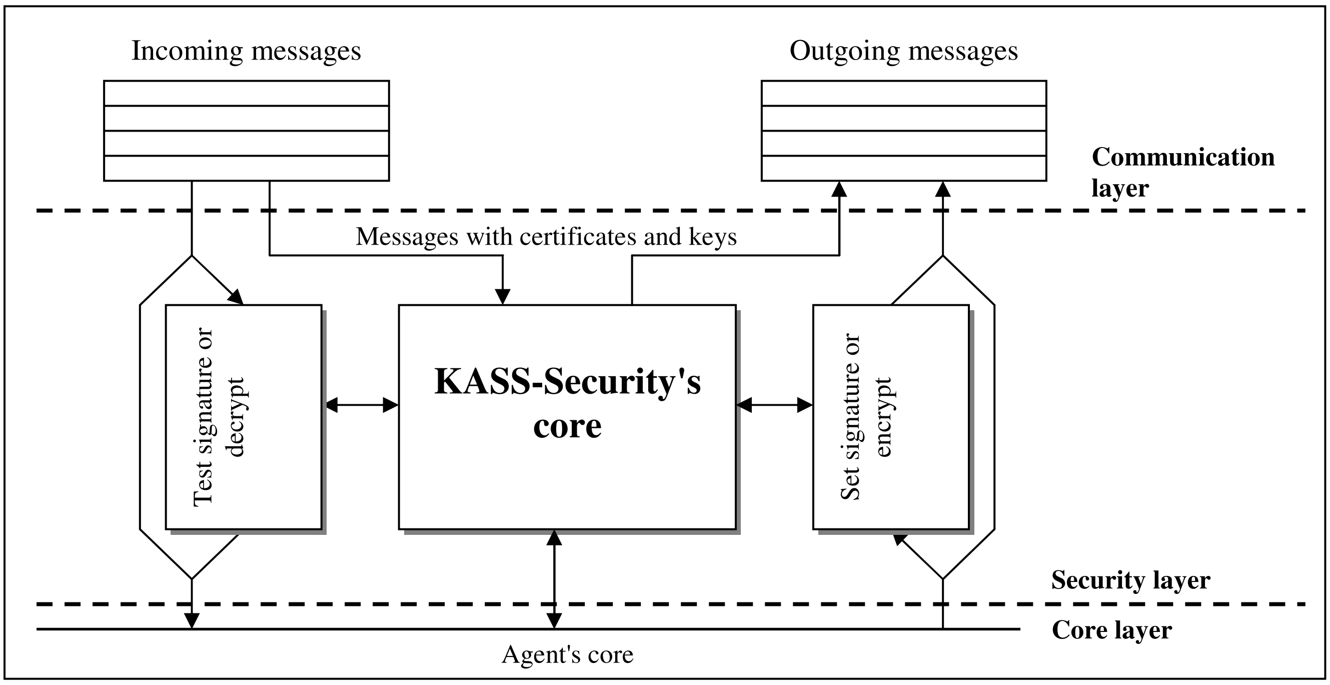
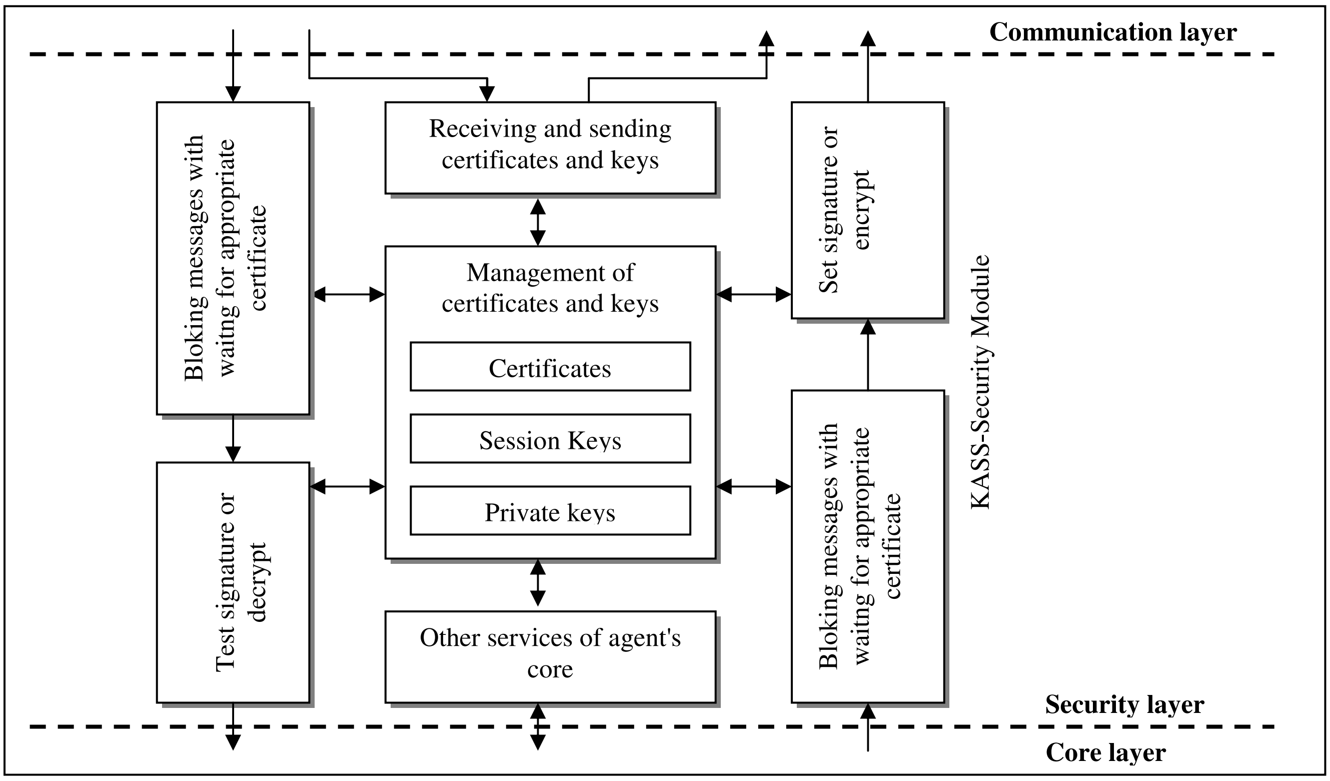


Figure 2 : Integration of security module to agent



Communication layer
Receiving and sending certificates and keys
Bloking messages with waitng for appropriate certificate
Set signature or encrypt
Management of certificates and keys
Certificates
Session Keys
Private keys
KASS-Security Module
Test signature or decrypt
Bloking messages with waitng for appropriate certificate
Other services of agent's core
Security layer
Core layer

Figure 3 : Security module layer

5. Conclusion

KASS system tries to avoid some disadvantages of the current security systems [9,10] such as failure of security functions during SCA inaccessibility, security uses of other communication channels and the control of MAS and security system from outside of the MAS. Proposed system has the following advantages: security can be included into already existing MAS, only parts of the message can be secured, system is usable in the area of mobile agents.

In addition to its advantages, this system has also several limits such as: as security regards, although there are an authentification of the agents and an access control, the level of protection is likely to be sophisticated. For example, we will be able to encrypt the transfer of the agents and the exchanged messages. Moreover, the agents profiting from this security system must be able to refuse any communication with not authenticated agents. It is also possible to extend the KASS system security module by adding it a decision unit which could prevent the security agent of the platform by a signal or a message [8, 11].

6. References

[1] Petr Novak et al., "X-Security Architecture in AgentCities". Czech Technical University in Prague. In: Agent Technology Competition 2003.

[2] Vlcek T., Zach J., "Considerations on Secure FIPA Compliant Agent Architecture". In: Proc. of IEEE/IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services.V.Marik, L.M.Camarinha-Matos, H. Afsarmanesh (Eds.). pp. 11-124, 2002.

[3] Foner, L.N., "A security architecture for multi-agent matchmaking". In: Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS96), pages 80-86. AAAI Press, 1996.

[4] Wong H. C., Sycara K., "Adding Security and Trust to Multi-Agent Systems." In: Proceedings of Autonomous Agents '99 (Workshop on Deception, Fraud and Trust in Agent Societies). May 1999, Seattle, Washington, pp. 149-161, 1999.

[5] FIPA 98, "Agent Security Management Specification". http://www.fipa.org/repository/obsoletespecs.html

[6] Burr W., Dodson D., Nazario, N., Polk, W.T., "Minimum Interoperability Specification for PKI Components", NIST Special Publication 800-15, 1998.

[7] Lyons-Burke K., "Federal Agency Use of Public Key Technology for Digital Signatures and Authentication", NIST Special Publication 800-25, 2000.

[8] Farah AbdelMajid BARIKA, "Vers un IDS Intelligent à base d'Agents Mobiles ", Mémoire de DEA en informatique, Institut Supérieur de Gestion de Tunis, Tunisie 2003.

[9] Chelliah Thirunavukkarasu, Tim Finin and James Mayfield, "Secret Agent-A Security Architecture for the KQML Agent Communication Language", CIKM'95 Intelligent Information Agents Workshop, Baltimore, December 1995.

[10] Katia P.Sycara, Timothy W.Finin. "Personal Security Agent : KQML-Based PKI". The Robotics Institute Carnegie Mellon University Pittsburgh, 1998, PA.15123.

[11] Bilel ELAYEB and Mohamed BEN AHMED, "Proposition d'un système de sécurisation d'agent à base de KQML", Mémoire de Mastère en informatique, Ecole Nationale des Sciences de l'Informatique, Tunisie 2004.