Communication Security in Multi-agent Systems

Petr Novák, Milan Rollo, Jiří Hodík, Tomáš Vlček

Gerstner Laboratory, Agent Technology Group
Department of Cybernetics, Faculty of Electrical Engineering,
Czech Technical University in Prague,
Technická 2, 166 27, Prague 6, Czech Republic
{novakpe,rollo,hodik,vlcek}@labe.felk.cvut.cz

Abstract. Both, research and application development in the area of multi-agent systems currently undertakes rapid expansion. In order to use multi-agent technology in real applications, it is inevitable to ensure security, integrity and authenticity of inter-agent communication. Various securitysystems, developed for different applications have been used in multi-agent system (MAS). Alternatively, MAS are designed with respect to the specific communication security requirements. This paper describes the architecture and implementation of the security (X-Security) system, which implemets authentication and secure communication among agents. System uses certification authority (CA) and ensures full cooperation of secured agents and already existing (unsecured) agents. Principle of the system integration into already existing MAS is described here as well as possibility of its usage for mobile agents. Paper deals as well with security mechanism's activity during inaccessibility of CA and possibility of CA's recreation. Security system was designed according the FIPA standards.

1   Introduction

In order to use multi-agent technology in real applications (such as industrial applications or e-business), it is inevitable to implement security communication among agents. This can be reached either by the message encryption (security against monitoring by undesirable side) or signing (assuring of message content's integrity). In some cases it is not necessary to secure whole message but only its parts.

There are a 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 later). The proposed approach attempts to avoid them and suggests a set of recommendations how to implement security in any programming language and any multi-agent platform.

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, while these new tasks enforce the requirements for these mediators much more thoroughness [1].

Existing FIPA specifications are proposed as open standards for agents' behavior and interactions. This paper highlights selected notions of trust and security used in ExtraPlanT MAS. Although FIPA specifications pertaining to security and network communication were started in 1998, there is still no complex and coherent document at disposal. Architectures proposed, e.g. [2] offers on different level the security and trust services defending MAS against corrupted naming (mapping or matchmaking) services, insecure communication channels, insecure agents delegations, lack of accountability, etc. [3]. 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 allows expansion of security into the area of mobile agents (agents migrating between platforms). It also defines a message extension for a new element that describes a form of the message security and tries to solve problems with inaccessibility of the central authority CA , which issues security certificates to particular agents. Agents use these certificates to prove their identity.

Proposed approach has been partially implemented as an extension of the multi-agent platform JADE [4] and tested within the AgentExchange project.

2   Security System Requirements

This paper describes design of security system prototype and implementation of its parts for multi-agent projects AgentExchange and ExPlanTech [5].

AgentExchange project aims at development of simple open trading environment. It is expected to operate as open MAS accessible to any agents. It is necessary to assure a certain degree of safety and trust within the community. ExPlanTech project (production planning and supply chain management) on the other hand presents closed MAS, but its agents can also run outside the agent platform (variety of databases) or special agent (for particular operating system) can be created. Platform integrates third-party information systems (e.g. ERP system) those company is running. In this case the security within the whole MAS is naturally required.

Security mechanisms, described in the only published FIPA security specification [6], are for mentioned systems unsuitable (possibility to secure only whole message not its part, ensures security for agents placed on agent platform only not for standalone agents, no support for mobile agents, etc.). Further on are described some additional requirements and dissimilarities from mentioned FIPA specification and outline of solution.

a) *Possibility to secure not only the whole ACL Message but only its part* is required as well. E.g. possibility to send a delegation (passed from one agent to another), sign certain part of text or data for storing 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 kind of requested data (e.g. from log file). This can be reached using the structure (class) containing not only carried text (signed or encrypted) but also additional in-

formation concerning security (type of security action, created signature, identification of used key). On the receiving side, this structure (class) is processed and original text obtained.

b) *Preferably not to bind security support tightly into the agent platform* because agents can run on different platforms (without this type of security) or even without 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.

c) *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 very 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 list of security algorithms (and public keys) [7], 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. This is suitable for mobile agents too [8, 9].

d) *All private keys and other security related data have to be available to their owner only*. Data have not to 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.

3   X-Security Prototype

In our proposed approach dedicated central authority is required to organize security mechanisms. This authority issues appropriate licenses – certificates. Agents use issued certificates to prove their identities and execute security related actions within the system [10]. Function of the central authority is in the proposed system exerted by the Security Certification Authority (SCA). Each agent using a security has to register its certificate with this 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 certificate. SCA can't guarantee validity of optional data, but can assure their constancy (originality) when providing other agents with the certificate. SCA signs 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 agent called testAgent:

```
certificate-ident            SCA_CERTIFICATE_1
    sca-ident                    (agent-identifier :name
                    sca@platform.net)
agent-ident            (agent-identifier :name testAgent@platform.net)
time-from            Wed Jan 01 00:00:00 CET 2003
time-to              Wed Dec 31 23:59:59 CET 2003
security-level       VISITOR
key-description
ident                SIGN_1
time-from    Wed Jan 01 00:00:00 CET 2003
time-to      Wed Dec 31 23:59:59 CET 2003
type                 public-key
key-param    SHAwithDSA/1024
key-value    56A7ED89C2………6AC54DF983
key-description
ident                CRYPT_1
time-from    Wed Jan 01 00:00:00 CET 2003
time-to      Wed Dec 31 23:59:59 CET 2003
type                 public-key
key-param    RSA/1024
key-value    5A234DC82B………85329B76DC
```

3.2   Integration of Security into Massage

In proposed system agents communicate using ACL Messages according to the FIPA standard [11]. Message is extended to contain a new slot called X-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          (FIPA-SL0)
      :content („Text to be signed")
      :X-Security   (    :type SIGN
    :signature 48A7………20AD
      :certificate-ident SCA_CERTIFICATE_1
      :key-ident SIGN_1 ) )
```

Items of the X-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          (FIPA-SL0)
        :content („28AD………7BA4")
        :X-Security   (      :type CRYPT
        :certificate-ident SCA_CERTIFICATE_1
        :key-ident CRYPT_1 ) )
```

X-Security slot items now inform that message content is encrypted by public key CRYPT_1 stored in certificate SCA_CERTIFICATE_1.

Analogously it is possible as well to secure only parts of the content.

3.3   Description of SCA's Activity

Common security system (completely) fails when SCA (or similar central authority) is inaccessible. 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 agent is allowed to ask for it directly 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.

Described approach also allows using the 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 verification when 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 the case of encrypting a huge amount of data or in the case of frequent communication between two agents the usage of asymmetric keys is not apposite because of the time and computational consumption. 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 symmentric 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 security module.

3.5   Common Agent Key Replacement

After a certain period of time or when the suspicion on the key misusage occurs, an agent is allowed to generate new keys. Immediately after generating them the 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 X-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 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 security level. This can happen when an agent with 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 certificates' ID's 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 change of its certificates SCA allows agents to update their certificates.

3.7   SCA Inaccessibility

As was already stated each of agents has its own certificate and these certificates are registered with SCA. During the temporary SCA inaccessibility agents are allowed to provide their certificates each other. In this time agents can't register new certificates but already existing security is not influenced. But permanent lost of SCA causes troubles for the communication security. This problem can be solved either by recovery SCA from backup or creating a brand new one.

There could be second SCA present in the agents' community. This SCA is only a backup and synchronizes its database with the main one for the case of losing the main SCA. In other case certain number of backup SCA is required. When the main SCA is lost the first backup becomes the main one and new backup SCA is created to complete number of backups. Another possible variant is that all of SCAs are active and can register common agents' certificates. All active SCAs synchronize their databases.

In a special situation of the MAS crash no SCA and only some of common agents could stay active. Then there is no backup of SCA that could be used for its recovery. In such a case agents must be able to create new SCA. At first agents create new instance of 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. Common agents send their certificates confirmed by old SCA to the new SCA. SCA sends them their 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 group of mobile agents for example for creating other secured temporary agents. Mobile agents have to keep their older certificates that are relevant for their home platform.

4   Implementation

SCA is a standalone agent that does not influence common agents' communication. There is only requirement to start SCA as a first agent of the community. Distribution of agents on platform is on Figure 1.
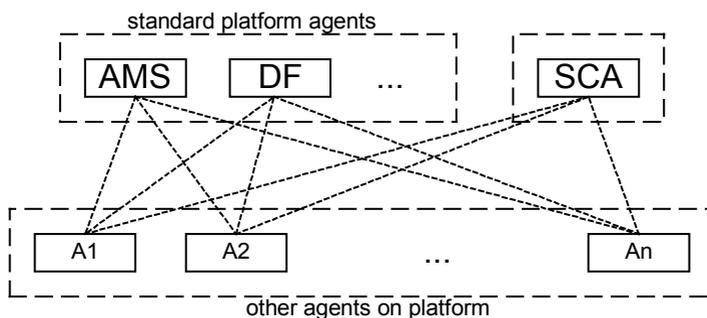

Fig. 1. Agent platform with Security Certification Authority

Security service is provided by security module that is placed between the agent's core and communication layer [12], 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 are passed directly to the agent's core.

The queue of outgoing messages contains messages created by security module (e.g. request for certificate) and messages created by the agent's core. The second ones are secured according to the requirements of the core. Agent core is allowed to restrictedly influence behavior of the security module. Inner structure of security module is shown on Figure 3.
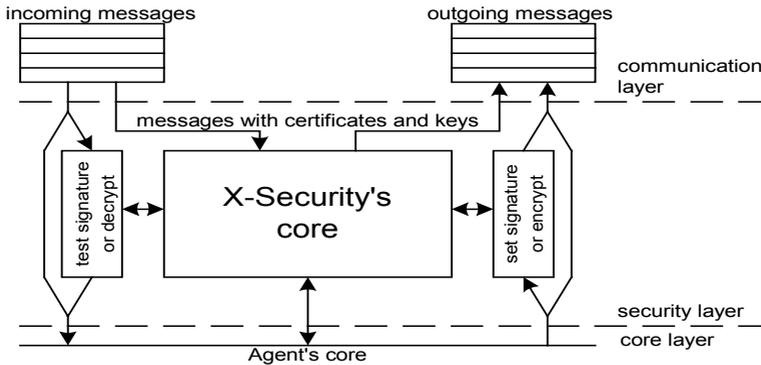
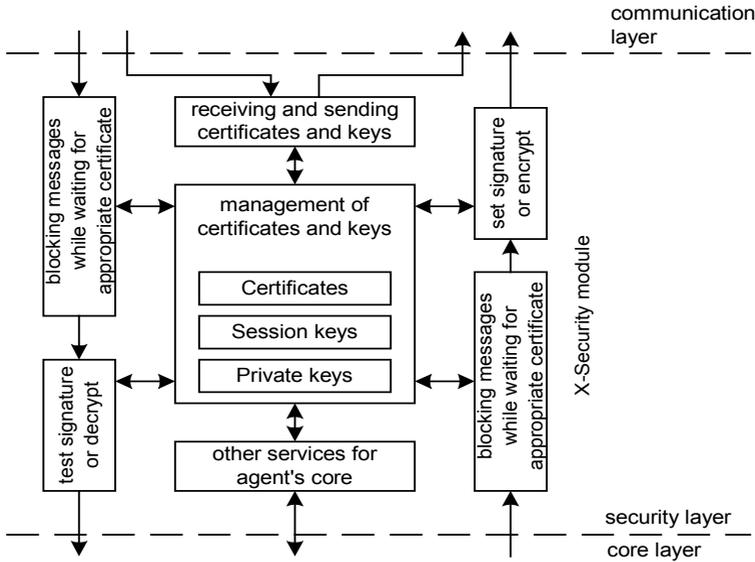**Fig. 2.** Integration of security module to agent

Fig. 3. Security module structure

Security module contains several individual units. One of them provides encrypting, decrypting, creating and checking the message signature [13]. Second one provides connection to SCA and exchanges certificates with other agents. Next unit 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. Last unit provides interface between security module and agent's core, which is necessary e.g. to configure the security or when partial encryption is required.

5   Experiments

Described security system has been included and tested within the Agent Exchange (AX) project. This project aims at developing a model of trading environment and a test-bed for modeling various market situations, investigation of market trends in environment of specific groups of traders' behaviors.

AX agent business community consists of Trading-agents (buy and sell commodities), Bank-agents (maintain commodity accounts), Exchange-agents (find matching orders to buy and sell), User-agents (lead the team of Trading-agents) and the Scenario-agents (influence the market according the pre-prepared plan). Each of these agents has to register its certificate including agent's role and competences (agent gains them at the creation time). The certificates are registered with SCA trusted by all community members. During the registration agent sends its username and password to SCA. According this data SCA sets up a security level in the registered certificate. When agent asks other one to perform any action, the second one proves competence of the first one by its certificate. Two simple cases of certificate exploitation are described bellow.

If Bank-agent wants to establish a new bank, it asks Central Trading Authority (CTA) for permission. CTA checks up security level in Bank-agent's certificate and if it is correct, CTA issues to Bank-agent appropriate license. This license allows Bank-agent to create and carry on new bank.

If a Trading-agent wants to create a new bank account, Bank-agent creates it and links agent's certificate with this account. From this time every attempt to access this account must contain signature. Bank-agent verifies the signature, using the certificate linked with this account and in the case of authenticity executes requested action.

Another MAS, in which described security system will be tested, is ExPlanTech, a production planning and supply chain management multi-agent system. There may be agents running outside the agent platform (variety of databases) and sending data to MAS in a secure way. Agents may also run on PDAs and other mobile devices for remote access to the system to set and verify orders as well. Every access to orders will be verified by agent's certificate.

6   Conclusion

Selected functions of the described system appropriate for the application were implemented in Java [14] programming language as an extension of JADE multi-agent platform. Developed library includes SCA agent and security module to be integrated into the common agents.

This system tries to avoid some disadvantages of the current security systems such as failure of security functions during SCA inaccessibility, security uses other communication channels then MAS and security system is controlled from outside of the MAS. Proposed system has 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.

This design represents a fundamental for universal security model which can be used for both research and application purposes.

References

[1]   Vlček 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 (BASYS 02). V.Marik, L.M.Camarinha-Matos, H.Afsarmanesh (Eds.). Kluwer Academic Publishers, Boston/Dordrecht/London (2002) 11-124

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

[3]   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). Seattle, Washington (1999) 149-161

[4]   JADE. http://jade.cselt.it, Java Agent DEvelopment Framework (2003)

[5]   Pěchouček, M., Říha, A., Vokřínek, J., Mařík, V., Pražma, V.: ExPlanTech - applying multi-agent systems in production planning. In: International Journal of Production Research, vol. 40, no. 15 (2002) 3681-3692

[6]   FIPA 98 http://www.fipa.org/repository/obsoletespecs.html Agent Security Management Specification (obsolete)

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

[8]   Jansen, W., Karygiannis, T.: Mobile Agent Security, NIST Special Publication 800-19 (1999)

[9]   Karnik, N.: Security in Mobile Agent Systems. Ph.D. Dissertation, Department of Computer Science, University of Minnesota (1998)

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

[11]   FIPA. http://www.fipa.org, Foundation for Intelligent Physical Agents (2003)

[12]   Bradshaw, J.M. (ed.): Software Agents, AAAI Press/MIT Press (2002)

[13]   Welschenbach, M.: *Cryptography in C and C++*. Springer-Verlag, Germany (2001)

[14]   Sun Microsystems. http://java.sun.com, Java Programming Language (2003)