

X-Security Architecture in AgentCities

Petr Novák, Milan Rollo, Michal Pichoušek, Tomáš Vlček

Gerstner Laboratory, Agent Technology Group
Department of Cybernetics, Czech Technical University in Prague,
Technická 2, 166 27, Prague 6, Czech Republic

{novakpe|rollo|pichouc|vlcek}@labe.felk.cvut.cz

Abstract: Both, research and application development in the area of multi-agent systems currently undertakes a rapid expansion. In order to use the multi-agent technology in real applications, it is inevitable to implement security, integrity and authenticity of inter-agent communication. Various security systems, 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 implements authentication and secure communication among agents. X-Security allows a secure, FIPA-compliant information exchange among the agent and it has been successfully integrated in the AgentCities environment. Currently X-Security is being integrated in ExPlanTech, a JADE-based production planning and supply-chain management system.

1 Introduction

In order to use multi-agent technology in real applications (such as industrial applications or e-business), it is inevitable to implement secure communication among agents. This can be reached either by the message *encryption* (security against monitoring by undesirable side) and/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 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 Chapter 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 break

down 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 disrupting of the communication paths between agents or attacking 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 to entities within a MAS required support to assure confidence in the distributed system and to prevent it from the identified security attacks. Focus is given to support security instruments for FIPA compliant architectures. 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 [1].

Existing FIPA specifications are proposed as open standards for agents' behavior and interactions. This paper highlights selected notions of trust and security required in the AgentExchange MAS. Although FIPA specifications pertaining to security and network communication were started in 1998, there is still no complex and coherent implementation available. 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.

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

Proposed approach has been partially implemented as an extension of the multi-agent platform JADE [4]. For JADE there is available a security plug-in

JADE-S, that adds security mechanisms related to the agent platform security [11]. It includes features such as user/agent authentication, authorization and secure communication inside an agent platform. It brings to MAS concept of allowed actions that can agent carry out on the platform – each component of the platform is owned by an authenticated user and is authorized by the platform administrator to perform only certain privileged actions. In currently existing MAS is e.g. possible to kill agent, move a malicious agent onto your main container, start a remote container, etc. Using JADE-S such actions can be carried out only by privileged agents. JADE-S is based on concept of Java security model and extends it for multi-agent systems. JADE-S enables the usage of the Secure Socket Layer (SSL) protocol, providing privacy and integrity for all intra-platform connections. Unlike in X-Security it's not possible to use this mechanism for inter-platform communication and only whole messages can be secured.

2 Security system requirements

AgentExchange (AX) is a simple, open trading environment developed for testing advanced negotiation protocols and auctioning strategies that may occur in complex agent communities. AX is ready to operate as open MAS accessible to any agents. In order to provide the AgentCities community with an environment for fair-trading, it is inevitable to implement a certain degree of safety and trust within the community of trading agents.

Owing to the openness of the AX community it is necessarily to allow a certain degree of interaction between the secured and unsecured agent. Security mechanisms, described in the only published FIPA security specification [5], do not allow securing 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 sum-up the security requirements and dissimilarities from the FIPA specification:

Partial message encryption: Possibility to secure not only the whole ACL Message but only its part is required 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 information 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.

Loose link with platform: Possibility not to bind security support tightly into the agent platform. FIPA-interoperable agents can run on different platforms (without the implemented type of security) or even can be stand-alone 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.

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. Each agent (its security module) has to register (with certain authority) a list of security algorithms (and public keys) [6] being involved. Agent with the intention of sending a secured message has to ask for a list of receiver-supported algorithms and use one of them for secure communication.

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.

Central Authority Inaccessibility: The security infrastructure shall enable agents operation 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 considered. Running multiple central authorities shall be also possible.

3 X-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). SCA generates its own certificate when starting up for the first time.

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 cannot 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 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              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.....85329E76DC
```

3.2 Integration of Security into Message

In the proposed system the agents communicate using ACL Messages according to the FIPA standard [8]. A 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`.

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
}

```

An example of message containing secured part of the content follows. 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 (FIPA-SL0)
:content
  (action
   (agent-identifier :name ... :address ... )
   (data-register
    :data (security-text
           :text „10,20,30,40,50“
           :security (security-description

```

```

:type SIGN
:signature 24A8.....D7C6
:certificate-ident
    SCA_CERTIFICATE_1
:key-dent 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 (FIPA-SL0)
  :content
    (action
      (agent-identifier :name ... :address ... )
      (data-request
        :data-type „type_of_requested_data“
        :password (security-text
          :text „84C7.....AD7B“
          :security (security-description
            :type CRYPT
            :certificate-ident
              SCA_CERTIFICATE_1
            :key-dent CRYPT_1 ) ) ) ) )

```

3.3 Description of SCA's activity

Common security system 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 the agent is allowed to ask for it directly 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 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 verifying 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 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 appropriate 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 misuse 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 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) certificates¹.

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 certifi-

¹ From security point of view this feature represents a security risk in case of compromise of 'old' certificate and therefore such mode is not recommended.

cate 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 the agents are allowed to provide their certificates one to the 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 lost of SCA causes troubles for the communication security. This problem can be solved either by recovery SCA from backup or by setting multiple (backup, secondary or caching) 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 number of backups. Alternatively, there is no predefined structure in the community of SCAs and the agents can register with any SCA accessible.

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 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 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 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's core is allowed to restrictedly influence behavior of the security module. Inner structure of security module is shown on Figure 3.

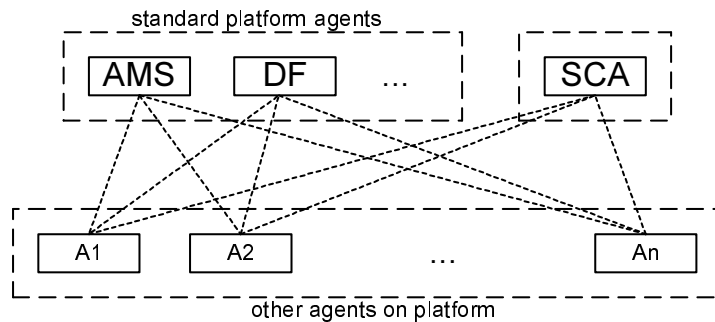


Fig. 1. Agent platform with Security Certification Authority

Security module contains several individual units. One of them provides encrypting, decrypting, creating and checking the message signature [9]. 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.

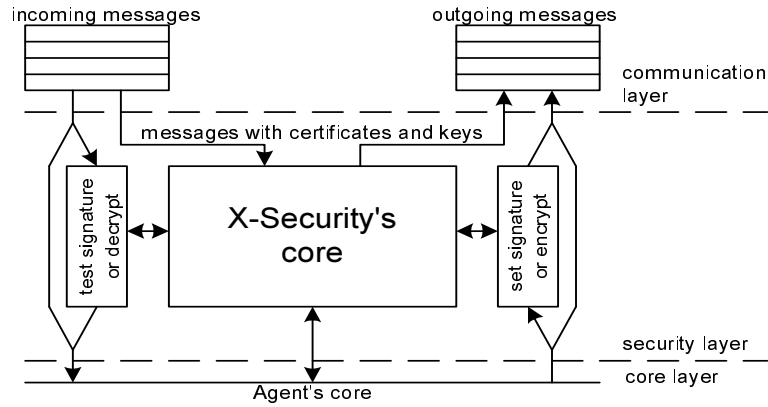


Fig. 2. Integration of security module to agent

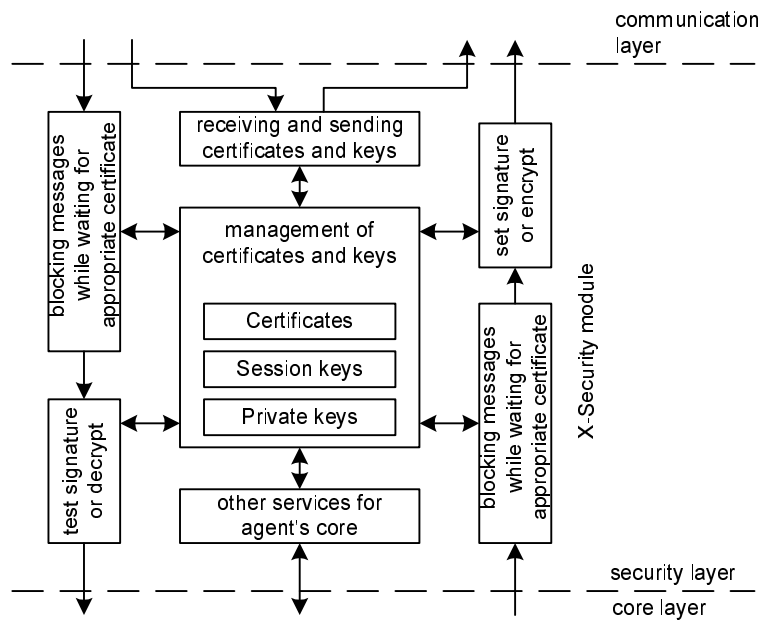


Fig. 3. Security module structure

5 Deployment

Described security system has been involved and tested within the AgentExchange (AX) system, a model of an agent trading environment and a test-bed for modeling various market situations, investigation of market trends in an en-

vironment of specific groups of traders' behaviors. The X-Security package² has been already deployed by several AgentCities nodes. The system has been acknowledged as the contribution to the AgentCities network by the 2nd main prize at the Agent Technology Competition 2003³.

Another MAS, in which the described security system will be tested, is ExPlanTech⁴, a production planning and supply chain management multi-agent system. By a community of agents, where each integrates (or simulates/monitors) a specific manufacturing device, we have managed to plan a wide range of very specific projects running in the manufacturing enterprise – the project has been supported by EC IST-Trial Project: IST-1999-20171 “*Exploitation of Agent-Based Production Planning using the ProPlanT Technology*”. The entire system has been developed in JADE and integrates the legacy systems in the factory (e.g. ERP systems, Material Resource Management System, etc.). Recently the ExPlanTech technology has been extended (due to support of the EUTIST-AMI (IST-2000- 28221) trial project - *Exploitation of the ProPlanT agent technology for extra-enterprise production planning*) for the extra-enterprise production planning activities. Extra-enterprise planning involves primarily an access to the production planning data from outside of the factory (via internet browsers, PDA devices or WAP-enabled phones) and automation of the supply-chain management related activities such as brokerage, negotiation and auctioning. For successful industrial applicability of the technology, confidentiality of the communicated data must be assured. Due to full JADE compliancy, the X-Security solution is being adopted in the ExPlanTech multi-agent system

6 Conclusion

Selected functions of the described system appropriate for the application were implemented in Java [10] programming language as an extension of JADE a FIPA-compliant multi-agent platform. Developed library includes SCA agent and security module to be integrated with the 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 than 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.

² available at <http://agents.felk.cvut.cz/security>

³ for more information see <http://www.agentcities.org/EUNET/Competition/>

⁴ for more information see at <http://agents.felk.cvut.cz/explantech> and <http://agents.felk.cvut.cz/extraplant>

The X-Security system has been acknowledged as the contribution to the AgentCities network by the 2nd main prize at the Agent Technology Competition 2003.

Acknowledgements

We thank the Agentcities.NET (IST-2000-28384) and IG CVUT (no. CTU0208513) for co-funding implementation of the X-Security project

References

1. Vlèek 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, pp. 11-124, 2002
2. 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.
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)*. May 1999, Seattle, Washington, pp. 149-161.
4. JADE. <http://jade.cselt.it>, Java Agent DEvelopment Framework (2003)
5. FIPA 98 <http://www.fipa.org/repository/obsoletespecs.html> Agent Security Management Specification (obsolete)
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. FIPA. <http://www.fipa.org>, Foundation for Intelligent Physical Agents (2003)
9. Welschenbach M. *Cryptography in C and C++*. Springer-Verlag, Germany (2001)
10. Sun Microsystems. <http://java.sun.com>, Java Programming Language (2003)
11. Vitaglione Giosuè: JADE Tutorial – Security Administrator Guide, TILAB, 2002