MagicNET: Secure Communication Methodology for Mobile Agents

Awais Shibli[1,2], Imran Yousaf[1], Kashif Dar[1], and Sead Muftic[1]

[1]*Department of Computer and Systems Sciences, KTH, Stockholm Sweden*

[2]*School of Electrical Engineering and Computer Science NUST, Islamabad Pakistan*

{ awais, imrany, kashifd }@ kth.se, sead@dsv.su.se

Abstract ---- Most of the current research and development results, dealing with authentication of mobile agents, describe solutions that address only agent-to-platform authentication. These solutions assign privileges to agents so that they can be executed and then, by using the same privileges, also to communicate with other agents running on the same platform. They do not address broader agent-to-agent communication security requirements. Moreover, communication protocols are not based on any standards, what increases the possibilities of communications between benign and malicious agents. In this paper we describe agent-to-agent secure communication methodology that guarantees authenticated, authorized and confidential communication between agents. We use FIPA ACL standard for effective and interoperable communication in our agent–based system.

***Keywords*— Mobile Agents, Secure Communication, Authentication of Mobile Agents, Agent-to-agent Communication**

1. INTRODUCTION

Careful analysis of mobile agents security research results created during the last decade reveals that significant efforts have been made to solve threats that originate from malicious agents attacking an agent platform or a malicious agent platform attacking hosted agents [1,2,3]. There has been little effort to address threats that stem from malicious mobile agents that may effects benign agents during communication between them. NIST in its report categorizes these threats as agent-to-agent threats [4]. Malicious agents can masquerade, i.e. act on behalf of another agent and then exchange rogue information, which can be disastrous to overall security of the system. Alternatively, malicious agents can repudiate undesirable actions and can get unauthorized access to critical resources of another communicating agent, for instance agent's baggage, which might contain some sensitive information.

There is a number of agent systems that all use simple communication mechanisms, i.e. they usually adopt an agent-based architecture based on a simple mobile agent execution scenario. According to such architectures, mobile agents are launched from a host machine, called *Agent Home*. They traverse predefined hosts specified in their route, execute remotely, and eventually return to *Agent Home.* This architecture bypasses the issues of agent-to-agent communication and therefore does not address the issues of threats to a mobile agent originating from other, potentially malicious agents, since mobile agents are not required to communicate with other mobile agents during their execution at remote hosts. However, it significantly deprives various mobile agent-based applications from benefits of mobile agents' paradigm. Mobile agents, being, "social" entities, communicate with each other in order to achieve better performance and goal-related benefits. For example, inter-agents communication is essential when an operation is distributed between different agents in order to enhance system throughput through parallelism. In that situation, different agents need to share each others' intermediate processing results, regardless of the agent platform on which they are executing. Agents may also collaborate with other agents in order to accomplish complex tasks. Collaboration may be with static agents (located at agent platforms), with the members of agent's own team, or remotely, with other independent agents executing at other agent servers (agent platforms). Therefore, mobile agents' collaboration is an essential aspect of every mobile agents system, so that the prerequisite for their effective and secure collaboration is to mitigate all potential security threats for mobile agents' communication. Among those threats, masquerading, repudiation, denial of service, and unauthorized access are of major concern [4].

In addition to having a secure communication, it is also important to have meaningful communication among mobile agents. Therefore, there is a need for a standard and predefined format of communication messages acceptable and followed by all vendors developing variety of different agent–based applications. In other words, there should be a common language based on shared vocabulary to be used in building different applications. The Foundation for Intelligent Physical Agents (FIPA) has provided an *Abstract Architecture Specification* for multi-agents system. In those specifications, FIPA has described standard message structure, message transport protocol, and message validity requirements [5]. FIPA specifications are known as Agents Communications Language (ACL). However, there is no specification of strong security for mobile agent communications.

In this paper we present a methodology that can be used by any mobile agent system for agent-to-agent authentication and secure communication. We have defined the format of messages and we used standardized solution for a complete set of security services needed by agents to securely communicate with each other. We have used our existing mobile agent system, MagicNET (described in section 3) for prototype implementation of the proposed methodology. Due to the space limitation, we will only mention the components, roles and agent platform architecture related to our proposed methodology, not the complete description of the MagicNET system.

The rest of the paper is structured as follows: section 2 highlights the related work in this area. Section 3 gives brief introduction of the MagicNET system along with detail description of secure methodology for mobile agents' communication. Section 4 concludes the paper, while section 5 suggests potential future research and development directions.

2. RELATED WORK AND STANDARDS

Neeran M . Karnik et.al [6] in their paper "*Ajanta Mobile Agent System*" proposed RMI interfaces for agents communication. Agents communicate with one another on a single platform or with remote agents on another platform. In the case of remote communications, some mechanisms for remote communication are necessary. Ajanta agents for this purpose use RMI interface (with enhanced security features). So called proxy interposition concept is used, where a proxy module is located between an agent and the outside object during communication. All incoming RMI invocations are intercepted by the proxy. The caller object (external object) can also authenticate itself by providing authentication data. Authentication mechanism is based on a challenge–response protocol. However, their work is not compliant with FIPA specifications.

Yuh-Jong Hu [7] highlighted security requirements for agents' delegation, with authentication and authorization in a multi-agent environment. They proposed an agent-oriented PKI for identification, authorization, and trust management. They also proposed different delegation mechanisms, such as threshold, chain-ruled, and conditional. That enabled them to enhance communicative acts in FIPA ACL. Although they verify certificates during delegation assignment, they do not provide certificate revocation process after delegation.

Varadharajan and Foster [8] proposed a model that supports delegation of privileges as agents move from one host to another. Delegation is a temporary responsibility that permits a child agent or cloned agent to act on behalf of the delegator agent. They address certain aspects, like verification if the delegator has actually transferred privileges to the delegated agent and whether it is the delegated agent that is making particular request.

Mobile Agents System Interoperability Facility [9], (MASIF) is a standard that defines the interoperability between different agent systems. First, MASIF describes the process of agents management that specifies how system administrators create, suspend, resume, or terminate agents. Second, it facilitates migration of mobile agents from one platform to another. Third, it helps agents and agent systems to identify each other by using well–defined agents' and agent system's namespaces. Fourth, it specifies the types of agent systems and standardizes their location syntax, so that agent systems can easily locate each other. However, MASIF does not describe formats required to build communication messages for mobile agents.

FIPA is an international organization which is committed to promote the technology of intelligent agents by developing specifications that support interoperability between agents and agent–based applications. Like MASIF, FIPA also addresses agent management system and agents' migration. In addition to previous developments, FIPA introduced an *Agent Communication Language (ACL)* for agents' communication. An FIPA ACL message contains a set of one or more message parameters. The exact parameters needed for effective agent communication vary according to the situation. The only mandatory parameter in all ACL messages is the *performative* that describes the purpose of communication. However, it is expected that most ACL messages will also contain identification of the sender and the receiver and an indicator of the content of a message. Specific implementations are free to include user–defined message parameters. In order to provide transport level security for agents' communications, FIPA specified a special parameter known as *envelope parameter* for an ACL message [5].

Finally, PKCS7 standard specifies the syntax of any cryptographic data (such as a digital signature or digital envelope) associated with a message [10]. The purpose of the PKCS7 is to provide a standard syntax and a platform-independent representation of the cryptographic data. The standard defines five different message types for different use cases. Among all these data types, PKCS7 `SignedAndEnvelopedData` type provides data confidentiality and integrity together with sender's and receiver's authenticity.

3. MAGICNET SYSTEM

MagicNET, which stands for *Mobile Agents Intelligent Community Network*, is a mobile agents system designed and developed in the Network Security Lab at ICT/KTH. MagicNET provides all infrastructural and functional components for research and development of secure mobile agents, such as, support for building secure and trusted mobile agents. It provides agents repository (agents' store), mobile agents' servers (for their execution), mobile agents' management stations, and security servers (IDMS server, CA server, etc). MagicNET has conceptually structured the overall system into three functional areas: a) *Agents Creation Area,* where agents are being created, validated, appraised, and published; b) *Agents Deployment Area,* where agents are retrieved and XACML polices are created for specific local domains; and c) *Agents Execution Area* which contains actual runtime components (physical network) for execution of

agents. Agents traverse the network and perform their tasks in the agents execution area.

Mobile agents execute in the agents execution area and therefore also communicate with other agents. Therefore, in this paper we will only focus on execution area. The next section explains components and operations of the system used for secure communications between agents.

3.1 Secure Communications Methodology

In this section we will describe components, operations, and authentication protocol (these are the different elements of our secure methodology) along with agent platform architecture used to achieve secure communications for mobile agents.

3.1.1 Components

a) Management Station Management station is a component of the system used by mobile agents' manager (agent owner) to perform various management functions with mobile agents. It provides GUI to the agents' manager to perform various tasks, i.e. launching of agents, communicating with agents, and receiving reports from agents. Agent Owner adopts different mobile agents at management station using a subsystem of MagicNET, explained in [14]. We will not go into details of adoption due to space limitation. We assume that Agent Owner at management station has already adopted the required agents and agents are ready to be launched in the network. Therefore, management station is the starting point of agents' execution.

b) MagicNET Runtime Node (Agent Platform) In any typical mobile agents–based application, mobile agents in different teams are launched from Agent Home (starting point of agent execution, in our case it is management station) into a network in order to perform designated tasks at remote hosts. Our secure communication methodology can be applied to any mobile agent–based system. We use the application mentioned in [12] as an example in this paper, where mobile agents are used for different tasks related to Intrusion Detection System. They perform their designated tasks at remote hosts (specified in the agents' route) and bring back their results to the management station.

Each remote host in the network is equipped with MagicNET Agent platform; it is called runtime node. Agent platform provides all necessary basic services for mobile agents execution, such as communication, registration, transport, agent-to-platform authentication and discovery services. Each runtime node possesses its own certificate issued by a local Certification Authority (LCA).

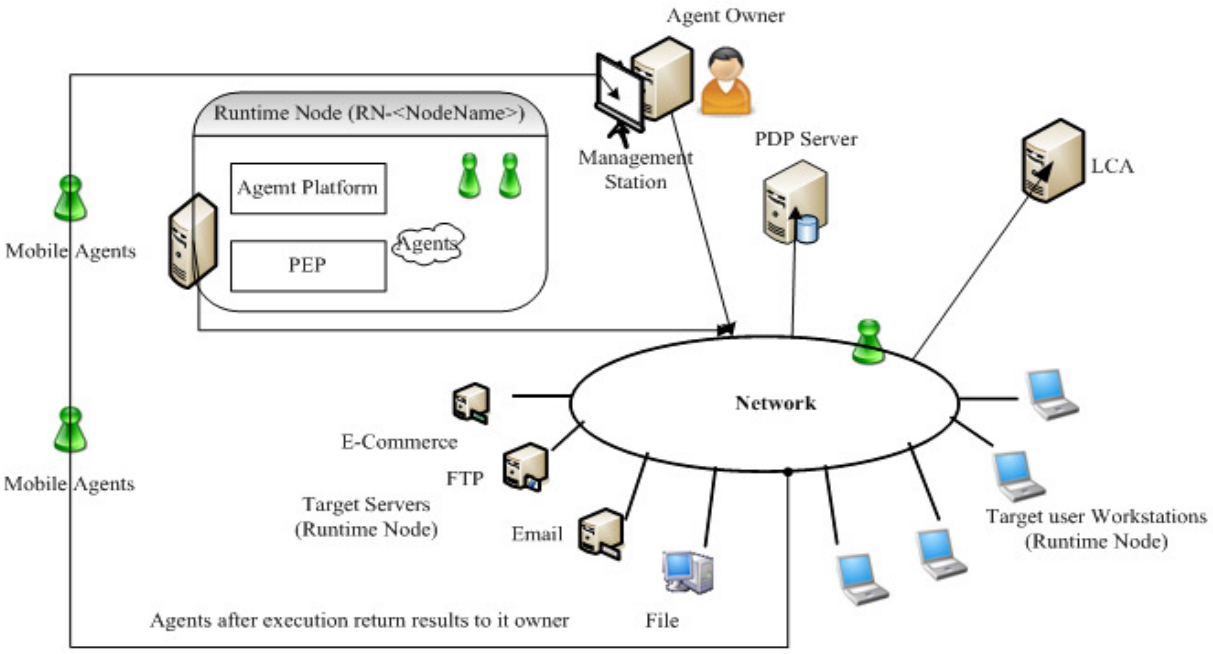

Figure 1: MagicNET Components (Network Structure)

The above diagram shows network structure of the MagicNET system. *Agent Owner* launches the agent and mobile agent starts travelling the network. The network may contain different kinds of *runtime nodes* to which a single mobile agent can travel. This mobile agent, if needed, can interacts with mobile agents residing at other *runtime nodes.* Every *runtime node* communicates with the Policy Decision Point (PDP) server, in order to get decisions regarding access rights and with LCA in order to get certificates.

c) Policy Enforcement Point (PEP) Policy Enforcement Point (PEP) [11] is the system entity that enforces access control by making decision requests and by enforcing authorization decisions. In our system, PEP at each runtime node, authenticates incoming agents and also agent-to-agent

communication requests by authenticating the communication initiator agent. Any mobile agent residing on another runtime node, which wants to communicate with an agent residing at the current runtime node, must own a valid SAML ticket issued by the PDP Server (explained in the next section) during initial agent authentication phase (explained in section 3.1.3 (c)), so that PEP uses the ticket to create mobile agent's authentication requests to the PDP.

d) Policy Decision Point (PDP) Server Policy Decision Point PDP [11] is the system component responsible for all security decisions concerning authentication and authorization in the system. In our system, PDP acts as an entity that collects credentials, generates assertions and issues SAML tickets to agents. In our system, PDP Server has been used to handle authentication and authorization requests coming from PEPs, to evaluate applicable policies, and to assert authorization or authentication decisions [11]. In this paper, we are using ticket generated for every agent TK_{Agent} by PDP in the Agent Authentication phase explained in section 3.1.3 (c). This SAML based ticket helps in authentication of every agent.

3.1.2 MagicNET Agent Platform Architecture

Figure 2 shows high level abstract architecture of the MagicNET agent platform, i.e. a hosting environment for mobile agents. Mobile agents migrate from one host to another and execute at remote hosts within the environment using supporting functions provided by mobile agents' platform. The second layer from the top shows the core services any mobile agent platform should have for mobile agents' execution. We will only describe the details of the top layer.

There are four main modules of each agent platform: *a) ACL module, b) Authentication module c) Encryption module,* and *d) Certificate Management module*. *ACL module* is responsible for construction of FIPA compliant ACL messages for agents' communication. *ACL module* also, in combination with *authentication module,* creates ACL messages for mutual authentication between two agents. Finally, in order to secure ACL messages exchanged between communicating agents, *Encryption module* is used, which envelops all the outgoing ACL messages and de-envelops every incoming message. *Certificate Management module* fetches certificates from a local CA server and provides services related to certificate management, like fetching certificates, renewing certificates, submitting certificate request, etc.

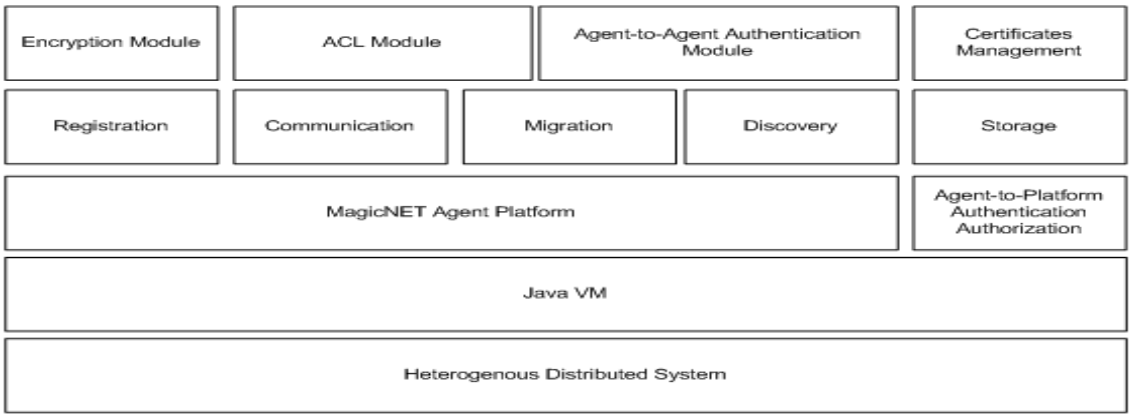

Figure 2: MagicNET Agent Platform Architecture

3.1.3 Operations

a) FIPA compliant ACL messages Creation ACL module uses SAGE ACL module [13] which facilitates creation of any FIPA compliant ACL message header and content parameters. SAGE ACL module is used to encode/decode ACL messages by using the required engine classes. We have incorporated those classes in our project. Table 1 provides short description of these classes, details of which can be found in [13]. ACL module has an `ACLMessageFactory (ACLMF)` class used to create all ACL messages. For example, if there is a need to generate ACL message for authentication purpose, then the authentication module will invoke `ACLMF` which first requests `ACLMEssageTemplate` class to create an ACL message template with default parameter values. Then, `ACLMF` instantiates `CFAgentAction` class to create ACL message content parameter of type `CFContent`, which is then loaded by `ACLMF` class into ACL message object.

It is important to mention that before sending ACL message to its destination, ACL message object must be encoded into an FIPA compliant SL string in order to provide message security and reduce network overhead. For that, `ACLMF` finally calls `ACLCodec` to encode ACL message object into FIPA compliant SL string. For this purpose `ACLCodec` inherently uses `SLTokenizer` to encode ACL content parameter into required FIPA compliant SL string and returns the complete ACL message to the `ACLMF` in the form of FIPA compliant SL string. Figure 3 shows the complete scenario of collaboration between these classes.

Table 1: ACL, Authentication, and PKCS7 Module classes

Module		Class Name	Description
ACL Module	Custom designed	ACLMessageFactory	Creates FIPA compliant ACL message.
		ACLMessageTemplate	Creates template of an ACL message with default parameter values.
	SAGE ACL Module	CFAgentAction	Creates content parameter for ACL message of type CFContent.
		CFContent	Extends CFObject (Content Facilitator) interface and stores ACL content parameter.
		SLTokenizer	Encode/decode ACL content parameter into FIPA SL string/CFContent.
		ACLCodec	Encode/decode ACL message into FIPA compliant SL string/ACL message object
Authentication		Authentication	Contains functions to create secure ACL messages for mutual authentication.
PKCS#7		PKCS7	Envelop/de-envelop ACL messages into signed and enveloped data/FIPA SL string.

b) Enveloping and De-enveloping ACL messages

Encryption Module supports enveloping and de-enveloping functions for any message. It uses PKCS7 `SignedAndEnvelopedData` type to secure the ACL message. Enveloping is done using `SignedAndEnvelopedData` PKCS7 package, comprising the following information: *signer info, content info*, and *recipients' info. Content info* contains information about the content, which in our case is an ACL message. *Signer info* contains information about the *runtime node* which signs the content i.e. ACL messages. *Recipient info* contains the information about the runtime nodes which will receive this package i.e. certificates of recipient runtime node.

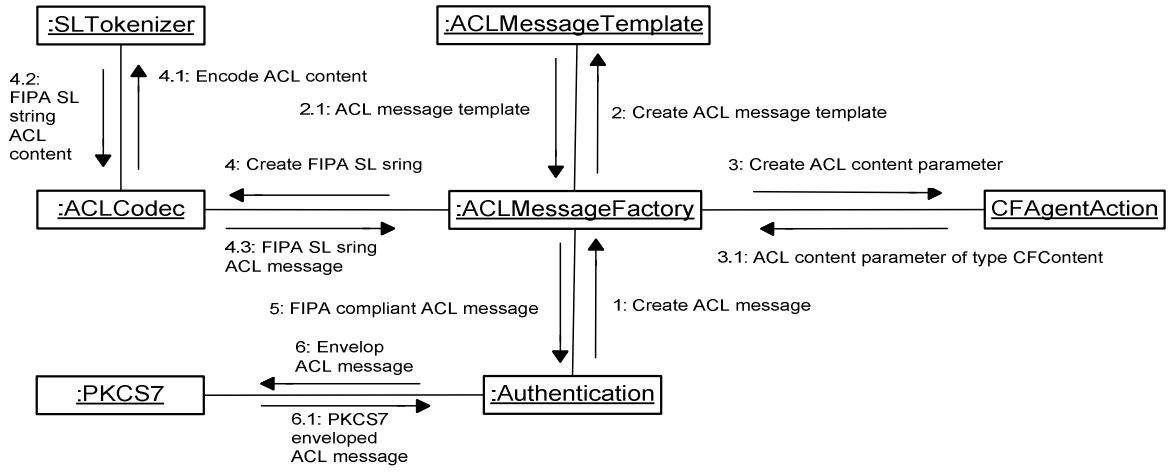

Figure 3: Interaction between different classes to create and envelope an ACL message.

c) ***Agent Authentication*** There are two types of authentications required for mobile agents: (a) Agent–to–platform authentication: when an agent arrives to the platform the first time, it authenticate itself in order to execute within agent platform; (b) Agent–to–agent authentication: when an agent wants to communicate with another agent executing on a same or remote agent platform.

The first type of agent authentication i.e. agent authentication by an agent platform for our MagicNET system, has been covered in [14]. According to it, agent authentication process involves three entities: *Agent Owner*, *Agent*, and *PDP Server*. The *Agent Owner* sends an authentication ticket request for a specific agent to a PDP in order to initiate the authentication procedure. This request is

$$AO_{tkReq.} = \{ C_{agent} \} \quad (1)$$

$$C_{agent} = \{H (agent_{c)}, S_{AC}, PK_{AC}\} \quad (2)$$

C_{agent} is the agent code, S_{AC} is the signature of *Agent Creator* of agent's code, and PK_{AC} is the public key certificate of *Agent Creator*. Once PDP receives AO_{tkReq}, it verifies its contents i.e. C_{Agent} . If the verification procedure is successful, i.e. C_{Agent} is valid, then PDP issues a SAML authentication ticket TK_{Agent} to the Agent Owner.

$TK_{Agent} = \{rand, IP_{PDP}, t_{exp}\}$ (3)

PDP caches this ticket in order later to authenticate the agent. This ticket is then used by the agent whenever it wants to execute at any agent platform. This ticket is presented to the PEP component at the runtime node in order to prove the authenticity of the agent. PEP then sends this ticket to PDP on order to verify its contents. If positive result is returned, then agent is considered to be an authentic agent. The details of this procedure have been published in [14].

3.1.4 Agent-to-Agent Authentication Protocol

The authentication protocol designed for agents' identity verification is based on the concept of mutual authentication, i.e. both communicating agents will authenticate each other. Suppose there are two agents, A and B, at two different *runtime nodes,* RN-2 and RN-2 respectively. The following are the secure communication requirements that must be ensured for each participating agent: prevent masquerading, disallow unauthorized access, avoid denial of service, and detect repudiation.

In order to meet these requirements for secure communication between agents, the following authentication protocol is designed: Figure 4 shows agent-to-agent authentication protocol.

1. Agent A generates a message containing four elements: ticket TK_{AgentA}, randomly generated agent's initialization ID_{AgentA} (when agent was registered with this host agent platform), distinguished name DN_{RN-1} of the runtime node RN−1 and agent's role R_{AgentB}. Since RN−2 can host multiple agents running simultaneously, that is why R_{AgentB} is needed to distinguish the agent that agent A wants to communicate to. This message is treated as 'content' in ACL message. This ACL message is then enveloped into PKCS7 SignedAndEnvelopedData package, $PKCS7_{A\{RN-1\}}$ in order to ensure message confidentiality and integrity. RN−1 then sends this message to RN−2.

   $PKCS7_{A\{RN-1\}} = ACL\{ TK_{AgentA}, ID_{AgentA}, R_{AgentB}, DN_{RN-1} \}$ (4)

2. RN−2, upon receiving the message, decrypts the package $PKCS7_{A\{RN-1\}}$. It then retrieves the content of the package i.e. ACL message. RN−2 checks the certificate and DN_{RN-1} in order to verify if the package originally came from the RN−1. Once the verification of RN−1 is completed, RN−2 retrieves the ticket TK_{AgentA}. In order to check the integrity and validity of this ticket, PEP_{RN-2} sends SAML authentication request to PDP: Req = { TK_{AgentA} }. PDP verifies if TK_{AgentA}'s parameters properly match and if so, returns an authentication response.

   Once, the identity of RN−1 and authenticity of the TK_{AgentA} have been verified, the Authentication Module at RN−2 sends back a PKCS7 package to the RN−1, having an ACL message. The contents in an ACL message are: pre initialization ticket of agent 'B' i.e. TK_{AgentB}, distinguished name of RN−2 (DN_{RN-2}), Agent A's random initialization ID_{AgentA} as received in the $PKCS7_{A\{RN-1\}}$ and Agent B's random initialization ID_{AgentB}. This package is denoted as $PKCS7_{B\{RN-2\}}$.

   $PKCS7_{B\{RN-2\}} = ACL\{ TK_{AgentB}, ID_{AgentA}, ID_{AgentB}, DN_{RN-2} \}$ (5)

3. RN−1 receives the package $PKCS7_{B\{RN-2\}}$ and decrypts it. Once the package is decrypted, it checks the certificate and DN_{RN-2} in order to verify whether the package originally came from RN−2. It then verifies the authenticity of the ticket TK_{AgentB} as explained in previous point (2). If the verification is successful, agents start the communication.

4. CONCLUSIONS

In this paper we have described a secure methodology for agent-to-agent secure communications. Using this methodology, mobile agent on two different agents' platforms can authenticate each other, before establishing any communication. FIPA ACL is chosen to formulate communication messages to provide interoperability between different vendor's applications. Before starting formal communication, two-way authentication methodology is designed and PKCS7 SignedAndEnvelopedData type is used to provide messages confidentiality and integrity.

Our methodology uses well–adopted security standards, like XACML, PKCS7, and SAML. We have evaluated our methodology against known security threats and we are confident that it can cater these threats; moreover, our methodology is designed in such a way that it can incorporate further future changes.

5. FUTURE WORK

There are number of research problems that still exist in this area. The first problem is related to *delegation of access rights,* which occurs when one agent authorizes other agents to perform some operations on its behalf. Delegation has a number of open research challenges, like reconstruction of the complete delegation chain and verification of proper authorization requests of an agent claiming on behalf of another agent.

Another interesting problem is control of access to sensitive information. An access control mechanism should be applied to prevent unauthorized access of the agent's baggage. So far, we do not know of any published results that addressed the issue of access control over an agent's baggage.

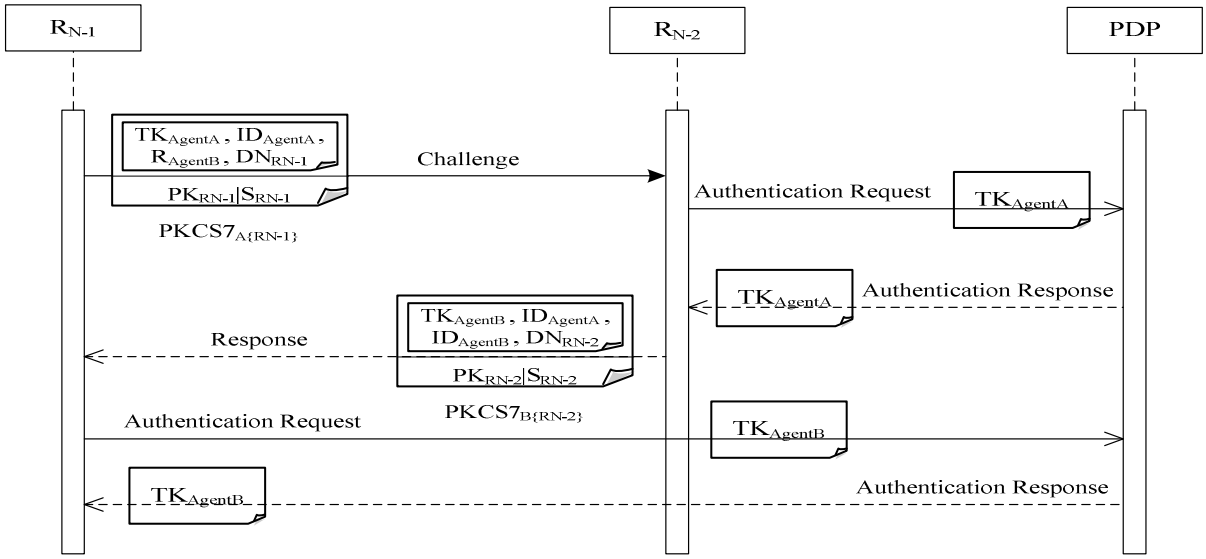

Figure 4 : Agent-to-agent Authentication Protocol

REFERENCES

1. Robles Sergi and Mir Joan, "*Implementation of Secure Architectures for Mobile Agents*", in MARISM-A [Book Section], Mobile Agents for Telecommunication Applications. - [s.l.] : Springer Berlin / Heidelberg, 2002. - Vol. 2521/2002. - 978-3-540-00021-1.
2. Bellavista Paolo [et al.], "*Security for Mobile Agents, Issues and Challenges*" [Book Section], Mobile Computing Handbook / book auth. Kelly Laurie, Ilyas Mohammad and Mahgoub Imad. - [s.l.] : CRC Press, 2004. - 0-84931-971-4.
3. Yi Cheng and Sead Muftic, "*A Comprehensive Security Infrastructure For Mobile Agents*", Licentiate Thesis / Department of Computer and System Sciences (DSV) ; Kungl Tekniska Högskolan (KTH). - Kista Sweden: DSV, 1997.
4. Wayne Jansen, Tom Karygiannis: "*Mobile Agent Security",* NIST Special Publication 800-19 (2000), Web: http://csrc.nist.gov/mobileagents/publication/sp800-19.pdf.
5. FIPA Specifications, "*Agent Security Management, FIPA ACL Message Structure, FIPA Ontology Service Specifications*", October 23, 1998, "FIPA Abstract Architecture Specifications", December 2002.
6. Karnik Neeran and Tripathi Anand, "*Security in the Ajanta Mobile Agent System*", [Journal] Software: Practice and Experience. - New York, NY, USA : John Wiley & Sons, Janaury 22, 2001. - 4 : Vol. 39. - pp. 301-329. - 0038-0644 .
7. Hu Yuh-Jong, "*Some Thoughts on Agent Trust and Delegation*", [Conference] // International Conference on Autonomous Agents. - Montreal, Canada : ACM NY, USA, 2001. - pp. 489 - 496. - 1-58113-326-X.
8. Poggi Agostino, Tomaiuolo Michele and Vitaglione Giosuè, "*A Security Infrastructure for Trust Management in Multi-agent Systems*", [Book Section] // Trusting Agents for Trusting Electronic Societies. - [s.l.] : Springer Berlin / Heidelberg, 2005. - Vol. 3577/2005. - 1611-3349.
9. Dejan Milojicic, Markus Breugst, Ingo Busse, John Campbell, "*MASIF, The OMG Mobile Agent System Interoperability Facility*", URL: http://www.hpl.hp.com/personal/Dejan_Milojicic/ma4.pdf.
10. An RSA Laboratories Technical Note, "*PKCS7: Cryptographic Message Syntax*", Standard Version 1.5, Revised November 1, 1993.
11. "*OASIS eXtensible Access Control Markup Language (XACML) Version 2.0*" [Specificiation], Feb 2005, web: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf
12. Awais. Shibli, Sead Muftic: "*Intrusion Detection and Prevention System Using Secure Mobile Agents*" [Conference] in Proceedings of the International Conference on Security and Cryptography, (pp. 76-82), Porto Portugal, July 2008.
13. NIIT, "*Open Source: Scalable, Fault Tolerant Agent Grooming Environment*", URL: http://sage.niit.edu.pk/SAGE_Components.htm, Accessed: March 2, 2009.
14. Awais Shibli, Alessandro Giambruno, Sead Muftic, Antonio Lioy, *"MagicNET: Security System for Development, Validation and Adoption of Mobile Agents"* [Conference] The 3rd IEEE International Conference on Network & System Security, Gold Coast Australia, October 19-21, 2009.