SECURE INTER-AGENT NEGOTIATION AND COMMUNICATION*

Mgr. Michal Laclavik, Dr. Ladislav Hluchy
Slovak Academy of Sciences
Dubravska cesta 9
Bratislava
Phone/Fax: +421 2 54771004
e-mail: laclavik.ui@savba.sk, hluchy.ui@savba.sk

Abstract. In this paper we are giving proposal for secure inter-agent communication and negotiation based on asymmetric cryptosystems and Certification Authorities. This security approach is common and widely used in Internet based systems and protocols such as ssh or https but never used in aglets agent systems for securing agent negotiation.

Keywords. Agents, Security, Negotiation, Communication

INTRODUCTION

Agents and agent systems are giving many possibilities for creating new generation applications. Communication and negotiation play important role in every agent application. Many people have worked on creating communications standards such as KQML [3] and ACL [4][5] and have integrated them to most of Agent Systems. Those standards are on very high level and supports many features. KQML standard is based on plain text messages with tags. If we want to use agents in real applications we have to solve also security of negotiation. Security in agent systems focuses mainly on migrating of agents and on security of agent environment. We did research about 50 agents systems and no one of them has solved security of communication, except for new version of Grashopper [9] Agent System. Even Grasshopper system focuses only on encrypting of messages but do not have support for agent certificates and authorities. We made proposal to solve this problem, based on standard methods such as asymmetric crypto-systems and certification authorities. PhD work Security of Agents [1] was good source, dealing with encrypting classes ElGamal [1] and secure migration of classes.
The similar idea was used in Agent Tcl [10] from Dartmouth University, where migrating agents are encrypted and authenticated using Pretty Good Privacy (PGP). Access restrictions are imposed on the agent based on its authenticated identity. Safe Tcl enforces the access restrictions. This method was use mainly because of TCL is scripting language so anyone could easy steal agent when migrating. TCL is one of first Multi-Agent Systems. Its secure migrating focuses us on researching and developing secure inter-agent communication and negotiation.

COMMUNICATION

Communication [3] [4] [5] is basically used for negotiating. All of multi-agent systems have to use kind of communication-negotiation protocol. Negotiation needs a communication protocol for agents to understand each other. There are many communication languages and protocols. Lot of agents systems are building their own ones. The most usable is getting KQML standard and also FIPA ACL standard. Both ACL and KQML is plain text communication. This way communication between agents can be stolen over the network. Almost no MASs solve problem of security of communication. We will focus on this area.

KQML

KQML or the Knowledge Query and Manipulation Language [3] is a language and protocol for exchanging information and knowledge. It is part of a larger effort, the ARPA Knowledge Sharing Effort, which is aimed at developing techniques and methodology for building large-scale knowledge bases, which are sharable and reusable. KQML is both a message format and a message-handling protocol to support run-time knowledge sharing among agents. KQML can be used as a language for an application program to interact with an intelligent system or for two or more intelligent systems to share knowledge in support of cooperative problem solving.

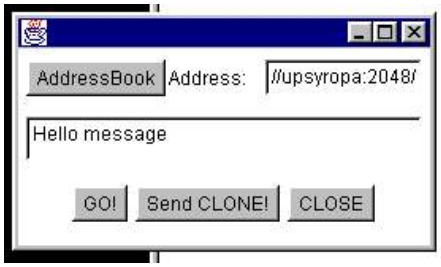
Agent Communication Language

The FIPA Agent Communication Language (ACL) [4] [5] is based on speech act theory: messages are actions, or communicative acts, as they are intended to perform some action by virtue of being sent. The specification consists of a set of message types and the description of their

* This work was supported by the Slovak Scientific Grant Agency within Research Project No. 2/7186/20

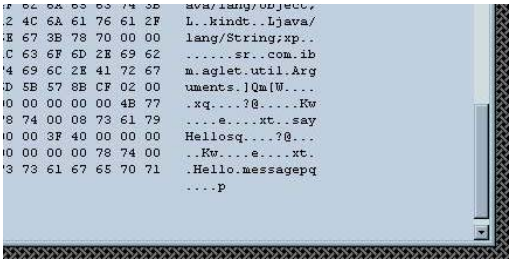
pragmatics, that is the effects on the mental attitudes of the sender and receiver agents. Every communicative act is described with both a narrative form and a formal semantics based on modal logic. The specifications include guidance to users who are already familiar with KQML in order to facilitate migration to the FIPA ACL.
The specification also provides the normative description of a set of high-level interaction protocols, including requesting an action, contract net and several kinds of auctions etc.

Example of unsecured communication in Aglets
As we already mentioned, security of communication is not solved in aglets. We are showing an example of stealing message across the network when an agent sends message to another agent. We installed ASDK [6] server on two machines. On third machine which is connected to the network on the same HUB we tried to steal the plain message and we succeed.
To be more explicit, we are sending only plain text message "Hello message" (Picture 1) instead of some KQML message.



Picture 1

This messages is received by other ASDK server:
To demonstrate stealing a message we used demo version of program NetSpy on the third machine.



Picture 2

On the picture 2 you can see text "Hello message" which was send and then received by second and third machine. To protect agent communication we are giving proposal of secure inter-agent communication.

PROPOSAL OF SECURE NEGOTIATION AND COMMUNICATION IN MAS

Security is very important issue in all systems. Many people can see and think that security is not solved yet in any Internet Based System. Security wholes and succeed hackers impacts are very often in the Internet world, because of programming mistakes or human failure. Theoretically we can say "Security is solved already" and it is true. The problem is that it has not been proposed and implemented yet in many systems. We can divide security to several levels [7]:

- Security of communication
- Security of System against outside impacts
- Access Rights
- Approving Users, Agents & Others

Security of Communication. We need to secure communication between citizens and our system. We can provide this by using SSL what is a standard encryption method used for example for the Internet Banking. Securing of communication in MAS is described below.

Security of System Against Outside Impacts. Choosing right and secure platform with installed security patches can solve most of those problems. In addition, some access restrictions must be set up.

Access Rights. Very important is securing against inside impacts. We need to define permissions for certain level for people, organizations etc. Also their communication with system has to be encrypted by public private key method.

Approving Users, Agents and Others - DPK Security Agent
Approving someone, who negotiate or communicate is very important. Even if we securing communication by public-private key method, we want to be sure that agent on the other site is the one what we think it is. For this purpose it is very important to make central or distributed database of agent public keys (DPK). Each Agent Place or each agent who has public key has to have this key stored in this DPK with its information. When new agent get created public key is generated and send to DPK by its creator with its information. DPK Security Agent can represent DPK. Each agent has standard method to access DPK agent by secure connection to get confirmation about public keys of other agents.

Securing of Inter-Agent Negotiation and Communication
KQML communication is not encrypted what means that anyone on the way to agent destination can read its communication. Because of this we have to encrypt communication. This can be possibly solved by public-private key encryption. Maybe some class, which implements this logic into agent communication, can be found. Using public-private key is standard way to protect any kind of data moving across network. If we will use public-private key encryption, we have to solve also generating those keys and remake standard communication functions of agent system. [7]
There are two kinds of cryptosystems: symmetric and asymmetric. Symmetric cryptosystems use the same key

(the secret key) to encrypt and decrypt a message, and asymmetric cryptosystems use one key (the public key) to encrypt a message and a different key (the private key) to decrypt it. Asymmetric cryptosystems are also called public key cryptosystems. Symmetric cryptosystems have a problem: how do you transport the secret key from the sender to the recipient securely and in a tamperproof fashion? If you could send the secret key securely, then, in theory, you wouldn't need the symmetric cryptosystem in the first place -- because you would simply use that secure channel to send your message. Frequently, trusted couriers are used as a solution to this problem. Another, more efficient and reliable solution is a public key cryptosystem, such as RSA, which is used in the popular security tool PGP.

RSA Encryption

The challenge of public-key cryptography is developing a system in which it is impossible to determine the private key. This is accomplished through the use of a one-way function. With a one-way function, it is relatively easy to compute a result given some input values. However, it is extremely difficult, nearly impossible, to determine the original values if you start with the result. In mathematical terms, given x, computing f(x) is easy, but given f(x), computing x is nearly impossible. The one-way function used in RSA is multiplication of prime numbers. It is easy to multiply two big prime numbers, but for most very large primes, it is extremely time-consuming to factor them. Public-key cryptography uses this function by building a cryptosystem, which uses two large primes to build the private key and the product of those primes to build the public key.

Algorithm:

- Find P and Q, two large (e.g., 1024-bit) prime numbers.
- Choose E such that E is less than PQ, and such that E and (P-1)(Q-1) are relatively prime, which means they have no prime factors in common. E does not have to be prime, but it must be odd. (P-1)(Q-1) can't be prime because it's an even number.
- Compute D such that (DE - 1) is evenly divisible by (P-1)(Q-1). Mathematicians write this as DE = 1 (mod (P-1)(Q-1)), and they call D the multiplicative inverse of E.
- The encryption function is encrypt(T) = (T^E) mod PQ, where T is the plaintext (a positive integer) and ^ indicates exponentiation.
- The decryption function is decrypt(C) = (C^D) mod PQ, where C is the cipher text (a positive integer) and ^ indicates exponentiation.

Your public key is the pair (PQ, E). Your private key is the number D (reveal it to no one). The product PQ is the modulus (often called N in the literature). E is the public exponent. D is the secret exponent.

You can publish your public key freely, because there are no known easy methods of calculating D, P, or Q given only (PQ, E) (your public key). If P and Q are each 1024 bits long, the sun will burn out before the most powerful computers presently in existence can factor your modulus into P and Q.

Proposed Secure Communication

Encrypted agent communication start as described: Agents will exchange their public keys by regular KQML requests, which are not encrypted yet. Each communication will start like following [7]:

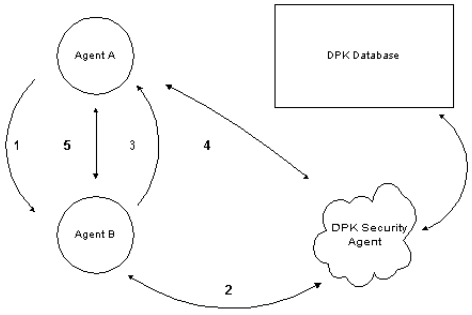
1. Request for communication from agent A to B with A's public key.
2. Encrypted request from agent B to DPK agent to get approval and other information about agent A and answer from DPK Agent.
3. Accepting request for communication and sending public key from agent B to A.
4. Encrypted request from agent A to DPK agent to get approval and other information about agent B and answer from DPK Agent.
5. All upcoming communication can by encrypted since this point.

Part of agent creation should be generating of those public-private pair key.

Agent migration securing can be solved by similar algorithm. Public and private keys of agent and destination place are used due to migration.

Agent Place Environment securing highly depends of used agent environment. If some Java based agent environment is used, Java Security Manager can be implemented. Still security levels and access priorities have to be defined.

Figure 3



IMPLEMENTING RSA AND DPK INTO AGLETS

Over 80% of recent MAS systems are based on Java that is the most appropriate language for Agents. We build our experiments on IBM Aglets [6] agent system and IBM JKQML class. However, the research and results will be useful for any other Java based agent system.

The following reasons justify Java Programming Language being ideal programming language suitable for our purposes:

- Java supports secure migration of classes.
- Interfaces to systems "speaking" KQML [3] are available for Java.
- Java is platform independent.
- SQL interfaces to databases are implemented.

There are development environments available for Java such as Borland JBuilder, Visual Café etc.
So implementation of our proposal for Aglets will be usable with little modifications in all other Java Based Agent System.

Why Do We Use Aglets?
Aglets were originally created by IBM Japan. Currently it is under GPL (general Public license) as Open Source Aglets.org project. Aglets are multi-agent system. An aglet [6] [2] is a Java object that can move from one host on the Internet to another. That is, an aglet that executes on one host can suddenly halt execution, dispatch to a remote host, and resume execution there. When the aglet moves, it takes along its program code as well as its state (data). A built-in security mechanism makes it safe for a computer to host entrusted aglets. Aglets are still under development, which is promising for implementing new features. We decide to use aglets for our experiments. Aglets do not support KQML but using IBM JKQML class can solve this. [8] Aglets are based on Java that is also very suitable for us.

We created following classes to secure agent communication:

Class Crypto. Supporting asymmetric cryptosystem RSA based on ElGamal [1] creating on this class automatically include public key and address of Security Agent. This is the reason why every agent using this class can get public keys from DPK by secure way

Class DPK. Database of Public Keys based on MySQL contains interface to this database, also supports functions for authorizing public keys, for registering etc.

Class SecurityAgent. Only this agent class is permitted to access DPK. Usually Agent Place should have one DPK and at least one Security Agent.

Described classes are not fully implemented but it works well for testing purposes. We assume that in time of presenting this paper whole implementation will be completed and verified.

CONCLUSION AND FUTURE WORK

As we already mentioned, no real application can be built based on an agent if security of communication is not solved. We are giving proposal and custom implementation to solve this problem. This technique, asymmetric cryptosystem based on RSA is known and widely used in other areas and convince us that bringing this technique into agent negotiation will be successful to secure communication. Agents have to communicate between different agent systems and KQML and ACL standards fit well this purpose. Those standards works with plain text messages so we have to secure them which we solved by this proposal.
In our feature work we will focus on verification of our implementation and on its use in some projects. Also we will try to find way how to connect DPK between each other and how to connect it with big certifications authorities as Verisign or Thawte
We intend to make Negotiation safer and usable for real applications.

REFERENCES

[1] PhD work Security of Agents, 1997

[2] Lange, D.: Programming Java Mobile Agents with Aglets. Addison-Wesley, 1998. Canada

[3] FIPA ACL Message Structure Specification, 2000

[4] KQML Website - http://www.cs.umbc.edu/kqml/

[5] FIPA: Foundation for Intelligent Physical Agents Geneva, Switzerland 1997
[6] IBM Aglets - http://www.trl.ibm.co.jp/aglets/

[7] Laclavik, M.: Negotiation and Communication in Agent Systems, 2001

[8] JKQML IBM -
http://www.alphaworks.ibm.com/tech/JKQML

[9] Grasshopper - http://www.grasshopper.de/

[10] TCL -
http://agent.cs.dartmouth.edu/general/agenttcl.html

BIOGRAPHIES

Michal Laclavik received the Mgr. (MSc.) degree in Computer Science and Physics from Faculty of Mathematics and Physics, Comenius University in Bratislava in 1999. His research focus on Negotiation and Communication in Multi-Agent Systems.

Ladislav Hluchy received the Dipl.Ing. (MSc.) degree from the Slovak Technical University Bratislava in 1975, and the CSc. (Ph.D) degree in computer science from Slovak Academy of Sciences in 1986. Hi is a member of IEEE Computer Society, IEEE Communication Society. His research interests include algorithms and methods for high performance computing.