Securing Mobile Agents for Electronic Commerce: An Experiment[†]

A.H.W. CHAN, K.M. WONG, T.Y. WONG, M.R. LYU
Department of Computer Science and Engineering, the Chinese University of Hong Kong, Hong Kong

Key words: Mobile agents, security, electronic commerce, information retrieval

Abstract: Mobile software agents are becoming a major trend of distributed systems in the next decade. Electronic commerce and information retrieval are two prospective applications of mobile agents. Nevertheless, security is a crucial concern for such systems. Attacks to agents by malicious hosts are the most challenging part of the problem unsolved. In this paper, a *Shopping Information Agent System (SIAS)* is built based on mobile agent technology. Possible security attacks by malicious hosts to agents in the system are discussed, and solutions to prevent these attacks are presented. Security of the solutions is analysed, and the performance overhead introduced is evaluated.

1. INTRODUCTION

Mobile agents are autonomous software agents that travel in a computer network to execute and perform tasks on different hosts for their owners. Several reasons for deploying mobile agents have been suggested, such as that [1]:

i. They reduce the network load;
ii. They overcome network latency;
iii. They encapsulate protocols;
iv. They execute asynchronously and autonomously;
v. They adapt dynamically;
vi. They are naturally heterogeneous; and

[†] The work described in this paper was supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region (Project No. CUHK4432/99E).

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35515-3_53

vii. They are robust and fault-tolerant.

A lot of mobile agent platforms have been developed around the world, such as Aglets [2] from IBM, Concordia [3] from Mitsubishi, and the Mole [4] from University of Stuttgart. Prospective applications of mobile agents include electronic commerce, information retrieval and network management. Nevertheless, security is one of the blocking factors of the development of these systems. The main unsolved security problem lies on the possible existence of malicious hosts that can manipulate the execution and data of agents [5].

In this paper, a *Shopping Information Agent System (SIAS)* is built using the Concordia architecture. The system is useful to collect and compare the prices of a set of products specified by users from different seller hosts in an electronic market. The security issues of the system are addressed and possible attacks by malicious hosts to the system are described. Solutions to protect the system against these attacks are devised and implemented.

The paper is organised in the following way: Section 1 (this section) is an introduction of the paper. Section 2 discusses the security issues of mobile agents in general, with focus on the problem of malicious hosts. Section 3 gives an overview of SIAS. Section 4 addresses the security problems and solutions of SIAS. An evaluation of the security solutions for SIAS is given in Section 5. Finally, Section 6 concludes the paper.

2. SECURITY ISSUES OF MOBILE AGENTS

Any distributed system is subject to security threats, so is a mobile agent system. Issues such as encryption, authorisation, authentication, non-repudiation should be addressed in a mobile agent system. Moreover, a secure mobile agent system must protect the hosts as well as the agents from being tampered by malicious parties.

2.1 Host Security

In a mobile agent system, hosts continuously receive agents and execute them. Hosts may not be sure where an agent comes from, and are at the risk of being damaged by malicious code or agents (Trojan horse attack). This problem can be effectively solved by strong authentication of code sources, verification of code integrity, and limiting the access rights of incoming agents to local resources of hosts, such that damages to hosts by malicious agents are limited to the resources available to agents. The solution is realised in the Java security model [6].

2.2 Agent Security

The main security challenge of mobile agent systems lies on the protection of agents. When an agent executes on a remote host, the host is likely to have access to all the data and code carried by the agent. If by chance a host is malicious and abuses the code or data of an agent, the privacy and secrecy of the agent and its owner would be at risk.

There can be seven types of attack by malicious hosts [5]:

i. Spying out and manipulation of code,
ii. Spying out and manipulation of data,
iii. Spying out and manipulation of control flow,
iv. Incorrect execution of code,
v. Masquerading of the host,
vi. Spying out and manipulation of interaction with other agents, and
vii. Returning wrong results of system calls to agents.

There are a number of solutions proposed to protect agents against malicious hosts [7], which can be divided into three streams:

i. *Establishing a closed network*: limiting the set of hosts among which agents travel, such that agents travel only to hosts that are trusted.
ii. *Agent tampering detection*: using specially designed state-appraisal functions to detect whether agent states have been changed maliciously during its travel.
iii. *Agent tampering prevention*: hiding from hosts the data possessed by agents and the functions to be computed by agents, by messing up code and data of agents, or using cryptographic techniques.

None of the proposed solutions solve the problem completely. A closed network effectively decreases the chance of an agent being attacked by unknown malicious hosts, however, it also limits the mobility and ability of agents. Agent tampering detection is possible but requires subsequent efforts to recover from attacks, and is not effective enough for agents that carry out critical missions. Agent tampering prevention would be most effective and useful, but is not yet feasible for all functions. There is no general methodology suggested to protect agents. Therefore, developers of mobile agent systems have to develop their own methodologies according to their own needs.

Apart from attacks by malicious hosts, it is also possible that an agent attacks another agent. However, this problem, when compared with the problem of malicious hosts, is less important, because the actions of a (malicious) agent to another agent can be effectively monitored and controlled by the host on which the agent runs, if the host is not malicious.

3. OVERVIEW OF SIAS

This section presents an overview of *SIAS*, the *Shopping Information Agent System*. SIAS is a web-based mobile agent system that provides users with information of products for sale in an electronic marketplace. It allows users to specify a set of products and the corresponding quantities they want to buy, and creates an agent to collect information about availability and price of these products from different hosts in the network. The path of the agent is determined before the agent is launched, according to the roster of hosts kept by the system. The design and implementation details of the system are described in the following subsections.

3.1 Design

SIAS is designed using the object-oriented paradigm because the concept of objects is useful to describe agents. There are three main types of objects in the system, namely Agents, Launch Servers and Database Servers. The object details and control flow of the system are described in this subsection.

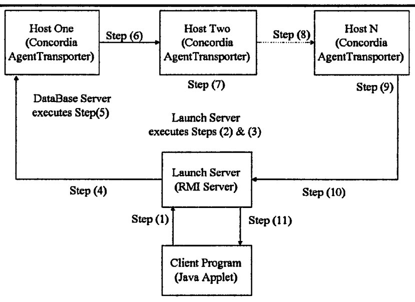
3.1.1 Object Description

The three objects are designed as follows:

i. The *Agent* object: it keeps a list of product identification numbers (IDs) and a list of the corresponding quantities specified by users. It is responsible to travel around the network and collect product information for users from different hosts.
ii. The *Launch Server* object: it is responsible for creating agents for users, sending the agents to the network, and receiving the agents when they finish visiting all the hosts specified in their itineraries.
iii. The *Database Server* object: it stores the information of products available at a particular host, (each host has its own instance of this object) and is responsible for retrieving required information for an agent when it arrives to the host.

3.1.2 Flow Description

When user makes a request for product information, an Agent is constructed with the product and quantity lists initialised properly by the Launch Server, and the agent will start its tour on the network. Whenever it reaches a host with a Database Server, it stays there, collects information of user-selected products, and then goes to another host. When it has visited all the hosts that are specified in its itinerary, it will calculate the lowest prices, and finally reports to user. The detailed control flow of the system is illustrated in Figure 1.


Host One (Concordia AgentTransporter)
Step (6)
Host Two (Concordia AgentTransporter)
Step (8)
Host N (Concordia AgentTransporter)
Step (7)
Step (9)
DataBase Server executes Step(5)
Launch Server executes Steps (2) & (3)
Launch Server (RMI Server)
Step (4)
Step (10)
Step (1)
Step (11)
Client Program (Java Applet)

Explanation of steps:

1. *Client program launches a request to the Launch Server object upon user input using Java Remote Method Invocation (RMI);*
2. *Launch Server creates an Agent object;*
3. *Launch Server initializes the agent with user-specified products and quantities, and the itinerary of agent;*
4. *Launch Server sends the agent to the network;*
5. *Database Server on Host One retrieves the required information for the incoming agent;*
6. *Agent goes to the next destination;*
7. *Database Server on Host Two repeats Step (5);*
8. *Agent goes to other hosts in the itinerary;*
9. *Database Server on each host repeats Step (5);*
10. *Launch Server receives the returning agent and calculates the cheapest purchasing combination;*
11. *Launch Server reports the cheapest purchasing combination to client program by Java RMI.*

Figure 1. Control flow of SIAS

Referring to Figure 1, there is an object on each host called AgentTransporter. This is introduced by the Concordia API, and it is responsible to listen for incoming agents. When an agent arrives, the AgentTransporter raises an event signal, and invokes the Database Server or Launch Server to handle the agent.

3.2 Implementation

SIAS is implemented using the Java programming language with the support of the Concordia API [3]. The choice of Concordia as the supporting API is primarily due to its ease to use and manipulate agent execution, so that malicious host actions can be simulated easily. The choice of Java as the programming language follows naturally since Concordia is Java-based.

4. SECURITY DESIGN OF SIAS

SIAS is a web-based system, attacks from the Web to the system are likely, and security is an important issue of the system design. Moreover, system security is of crucial importance to an application where money transaction is concerned. This section describes the security challenges of SIAS, and presents an approach to solve the problems.

4.1 Security Problems of SIAS

Both host security and agent security would be issues of SIAS. However, since SIAS is built on Java, which provides strong security mechanisms to protect hosts against malicious programs, the host security problem is very much simplified and solved. Therefore, this discussion focuses on the agent security part. Four possible types of attacks to agents that can compromise the security of the system are described here:

i. *Modification of query products*: The list of products specified by user is stored as the product ID list attribute of an Agent object, in plain text form. When an agent goes to a malicious host, the host can change its list of products to query about. When the agent later go to another host, the later host will respond to the changed products of query and report wrong information. This violates the integrity of the queries.
ii. *Modification of query quantities*: Similar to the modification of query products, a malicious host can change the quantities of products the agent want to query, violating the integrity of queries.
iii. *Spying out and modification of query results*: Agents carry query results also in plain text form. Therefore, when an agent goes to a malicious host, the malicious host can spy out and modify the results that the agent has collected from previous hosts in such a way that the changed results would favour the malicious host itself. This violates the confidentiality and integrity of query results.
iv. *Modification of itinerary of an agent*: The itinerary of an agent is accessible to hosts that have control over the Concordia platform where the agent lands and executes. When an agent goes to a malicious host, the malicious host can modify the path of the mobile agent so that the agent will go to a host not specified by user. This violates the authenticity requirement of the system.

The above are only a subset of possible attacks. There are other attacks such as replaying of query results and masquerading of hosts. However, these attacks are more complex, and require more efforts for both attack and defence. For the time being, only these four simple attacks are considered.

4.2 Solutions to the problems

After figuring out the above system vulnerabilities, mechanisms to protect the system against exploitation of these vulnerabilities should be implemented. As stated in Section 2, there is currently no good solution to mobile agent security in general. Therefore, application-specific security mechanisms should be devised.

A simple but original approach is developed to protect agents in SIAS against attacks from malicious host, based on cryptographic techniques. It is actually a mixed approach of the solutions discussed in Section 2.

i. *Closed network*: a new object, namely KeyServer, is introduced into the system, which provides a public key infrastructure for agents and hosts. Each agent or host should have a public key certificate registered to the key server for encryption use later on. This in effect establishes a closed network of hosts, among which agents are confined to travel.

ii. *Agent tampering prevention*: to protect query integrity, an agent can digitally sign its list of products and quantities using its private key, before it is launched. A host receiving the agent should verify the product and quantity lists with the signatures. Moreover, each host should encrypt the query results returned to the agent with the public key of the agent so that only the Launch Server can decrypt the query result. Furthermore, each host should digitally sign the query result it provides to ensure integrity and authenticity of the result returned.

iii. *Agent tampering detection*: the itinerary of an agent is a variable hidden by the Concordia system and normally not accessible. To protect the itinerary, the straightforward method is to encrypt it. However, this requires modification of the agent transporter of Concordia, which is not desirable.

The problem is worked around by making the itinerary an explicit attribute of an agent. When an agent arrives at a host, the host should read the itinerary of the agent, and encrypt the itinerary using its own private key to form a chain of encrypted itineraries (see Figure 2 IV). When the agent returns to the Launch Server, the Launch Server will decrypt the chain of encrypted itineraries using the public keys of the hosts to check the consistency of all itineraries with a copy of the original itinerary it saves before launching the agent. If a malicious host ever changes the itinerary of the agent, it is likely to be reflected in the encrypted itinerary chain and detected finally.

Figure 2 illustrates the changes introduced to SIAS for the security solutions described above, and Figure 3 illustrates the control flow of security-enhanced SIAS.

I. {Product ID list} *changed to*:
{Product ID list}•sig_A({Product ID list})

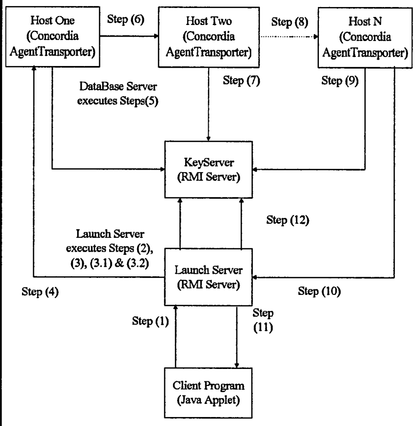
II. {Product Quantity list} *changed to*:
{Product Quantity list}•sig_A({Product Quantity list})

III. {Query result} *changed to*:
D_A({Query result}•sig_H({Query result}))

IV. *New attribute (chain of encrypted itineraries)*:
$E_{HN}(E_{H(N-1)}(\ldots E_{H2}(E_{H1}($Itinerary at Host 1) • Itinerary at Host 2) • ... Itinerary at Host N-1) •Itinerary at Host N)

<u>Key</u>
A: agent;
H: host;
H(k): k-th host visited by the agent;
sig_X(Y): digital signature of Y using the private key of X;
E_X(Y): the ciphertext of Y encrypted by the private key of X;
D_X(Y): ciphertext of Y encrypted by the public key of X.

Figure 2. Changes introduced to secure SIAS


Host One (Concordia AgentTransporter)
Step (6)
Host Two (Concordia AgentTransporter)
Step (8)
Host N (Concordia AgentTransporter)
DataBase Server executes Steps(5)
Step (7)
Step (9)
KeyServer (RMI Server)
Step (12)
Launch Server executes Steps (2), (3), (3.1) & (3.2)
Launch Server (RMI Server)
Step (4)
Step (10)
Step (1)
Step (11)
Client Program (Java Applet)

Explanation of additional / modified steps:
3.1. Launch Server generates a key pair for agent;
3.2. Launch Server signs the product and quantity lists for agents and registers the public key of agent to Key Server;
5. Database Server on Host One retrieves public key of agent from Key Server, and verify the signatures of product and quantity lists of agents. Then, the Database Server retrieves the required information for the incoming agent, signs the results using its own private key, and encrypt the results using the public key of agent, and also starts the chain of encrypted itineraries for agent;
11. Launch Server decrypts the query results, and verifies the signatures of the query results. It also detects change of agent itinerary by decrypting the chain of encrypted itineraries, and finally reports the cheapest purchasing combination to client program.
12. Launch Server deletes the public key entry of the finished agent from the key server.

Figure 3. Control flow of security-enhanced SIAS

5. EVALUATION OF THE SECURE SIAS

In this section, the security design implemented in Section 4 is evaluated. There are two aspects to evaluate. First, the security provided to SIAS by the additional measures is analysed. Then, the performance overhead introduced to the system by such measures is measured.

5.1 Security Analysis

The security of the additional measures lies mainly on the introduction of a key server that facilitates the use of public key cryptography. Assuming the key server and the communication channel are secure enough, a closed network can be built effectively.

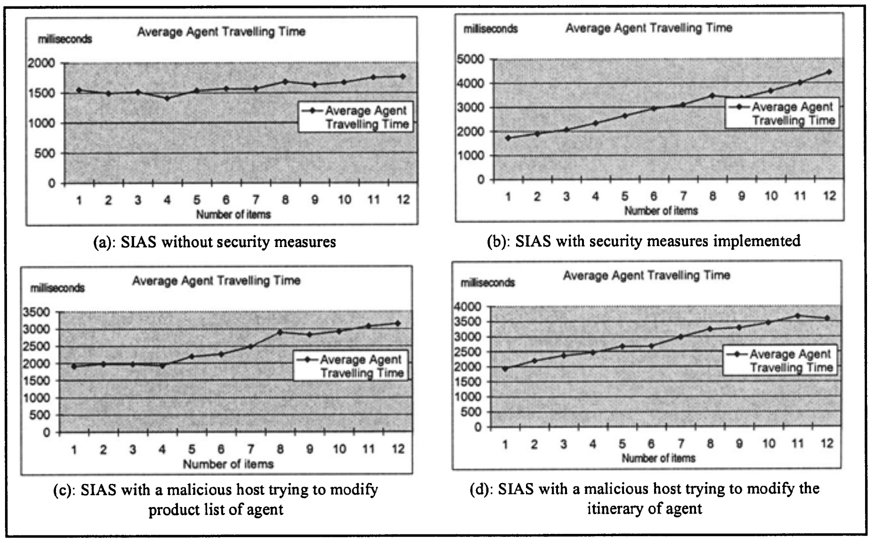
Furthermore, if the keys of agents are managed properly, the prevention of modification of the signed product and quantity lists of an agent by a malicious host is supported by the security of the underlying RSA encryption algorithm, in this case. The difficulty to break this algorithm is equivalent to that of the factoring problem.

Similarly, a malicious host would understand or modify the encrypted query results collected by an agent from another host at the same complexity. Therefore, integrity of queries, and confidentiality and integrity of query results, can be achieved by prevention of tampering.

For the detection of modification to itinerary of an agent by a malicious host, suppose there is only a single malicious host, out of N hosts, that wants to modify the itinerary of an agent. Since the encrypted itineraries are chained together, with one encapsulating another, the malicious host would need to fake all the (N-1) encrypted itineraries from other hosts to avoid being detected, which would be too complex to an ordinary attacker. Therefore, itinerary of the agent can be assured and authenticity achieved.

5.2 Performance measurement

The times for SIAS to launch a single agent before and after implementation of the security mechanisms described in Section 4 have been tested. Round trip times (RTTs) required for an agent to travel around an electronic market, consisting of three hosts, are measured under different situations. Queries of different sizes (number of product items) have been tested. RTTs measured are plotted against the query sizes in Figure 4.



(a): SIAS without security measures

(b): SIAS with security measures implemented

(c): SIAS with a malicious host trying to modify product list of agent

(d): SIAS with a malicious host trying to modify the itinerary of agent

Figure 4. Round trip time measurements for an agent in SIAS with different configurations

Figure 4(a) shows the results for the SIAS implementation without security measure implemented. The RTT increases very slightly with the size of query. The overhead introduced by each additional item is small, due to the small change in delay for database query with different query sizes.

On the other hand, Figure 4(b) shows that for the security-enhanced SIAS, the RTT increases very fast and linearly with the size of query. The overhead introduced by each additional item of query is about 250 milliseconds, which is about six times the overhead of the system without security measure. This significant overhead can be explained by the extensive use of the RSA algorithm to encrypt and decrypt each item, which is time consuming, especially when the key is long.

In addition to measuring the performance overhead introduced by the security measures, malicious hosts trying to modify the product list and itinerary of an agent in SIAS are simulated, and the overheads introduced by the actions of malicious hosts are measured. The results are reported in Figures 4(c) and 4(d). Both graphs show that an agent takes more time to travel around when there is attack from malicious host, compared with the measurements in Figure 4(a). The delay is quite significant (more than half of the original time). This suggests that the agent round trip time may also be used as a measure for tampering detection.

6. CONCLUSION

The technology of mobile agents and the problem of malicious hosts in a mobile agent system are discussed. SIAS is implemented as a sample application of mobile agents. Some security problems of malicious hosts in SIAS are addressed, and a primitive approach to protect the agents is developed. Security of the approach is analysed, and believed to be strong enough for domestic purpose. Performance overhead of the security enhancements is measured, showing a trade-off between performance and security for SIAS. This shows that it can take significant time for a malicious host to attack an agent.

REFERENCES

[1] Danny B. Lange and Mitsuru Oshima. "Seven Good Reasons for Mobile Agents", Communications of the ACM, p.88 - 89, 1999 Mar.

[2] "IBM Aglets Software Development Kit Homepage". http://www.trl.ibm.co.jp/aglets/

[3] "Concordia - Java Mobile Agent Technology". http://www.meitca.com/HSL/Projects/Concordia/

[4] "The Home of the Mole ". http://mole.informatik.uni-stuttgart.de/

[5] F. Hohl. "A Model of Attacks of Malicious Hosts Against Mobile Agents", Proceedings of the ECOOP Workshop on Distributed Object Security and 4th Workshop on Mobile Object Systems: Secure Internet Mobile Computations, p. 105 - 120, INRIA, France, 1998.

[6] "Java Security Architecture". http://java.sun.com/products//jdk/1.2/docs/guide/security/spec/security-specTOC.fm.html

[7] C. Tschudin. "Mobile Agent Security", Intelligent Information Agents: Agent Based Information Discovery and Management in the Internet, p. 431 - 446, Springer, 1999.