The Mobile Code Paradigm and Its Security Issues

Anthony Chan and *Michael R. Lyu*
Computer Science and Engineering Department
The Chinese University of Hong Kong
Shatin, Hong Kong

Abstract

Mobile code emerges as a new paradigm for developing non-interactive distributed applications. It requires less network communication compared with the conventional client/server paradigm, thus could reduce the bandwidth requirements of many applications, and ease network management in the ever-growing Internet. Active research is undertaken to bring the paradigm to realization, yet security concerns appear to be the major factor of its general acceptance.

In this paper, we examine qualitatively the security considerations and challenges in application development with the mobile code paradigm. We identify a simple but crucial security requirement for the general acceptance of the mobile code paradigm, and evaluate the current status of mobile code development in meeting this requirement. We find that the mobile agent approach is the most interesting and challenging branch of mobile code in the security context. Therefore, we built a simple agent-based information retrieval application, the *Traveling Information Agent* (TIA) system, and discuss the security issues of the system in particulars.

Keywords: Mobile Code, Security, Information Retrieval, Software Agent

1. Introduction

Distributed applications are applications that involve the coordination of two or more computers, geographically apart and connected by a physical network. Most distributed applications we see today, such as email, network news, file transfer, and Web browsing, are deploying the *client/server paradigm*. In the client/server paradigm, an application is divided into two processes, a client process running locally that asks for services and a server process on a remote site that gives services to the client. The client and server processes must communicate with each other in order to carry out their tasks successfully. Communication is done by means of message exchange. There are at least two problems with the client/server paradigm:

i. It has a high network bandwidth requirement due to the large number of messages exchanged.
ii. It usually requires users to response to computation results interactively, under different situations. Neither the client nor server would make decisions for users autonomously.

In view of the deficiencies of the client/server paradigm, the *mobile code paradigm* has been developed as an alternative approach for distributed application design. In the client/server paradigm, programs are geographically stationary, meaning that they do not move to another machine, and only run on the machines they reside on. (Therefore, the two processes must communicate continuously.) The mobile code paradigm, on the other hand, allows programs to be transferred among and executed on different computers. By allowing code to move between hosts, programs can interact on the same computer instead of over the network. Therefore, communication cost can be reduced.

Besides, the mobile programs can be designed to work on behalf of users autonomously. In this case, the mobile programs are called *mobile agents* [Whi98]. The autonomy of mobile agents allows users to delegate their tasks to the mobile agents, and not to stay continuously in front of the computer terminal.

The promises of the mobile code paradigm bring about active research in its realization. Supporting mobile code technology is also fast developing. Experimental mobile code systems that allow programs to move on a computer network have become available for download on the Internet [AS98]. However, these systems are not yet popular. Neither is the mobile code paradigm commonly adopted. Most researchers agree that the hurdle is security concerns [GBH98]. To gain general acceptance of the mobile code paradigm and systems, especially from *application developers*, there must be strong evidences for a certain security level for applications developed using such paradigm and systems.

In this paper, we try to collect such evidences. First, in Section 2, we review some of the foundation materials of the mobile code paradigm. We elaborate Ghezzi and Vigna's classification of mobile code paradigms [GV97], which is a collection of the *remote evaluation*, *code on demand* and *mobile agent* approaches. We demonstrate briefly their applications, and present our own classification of the supporting implementation technologies. In Section 3, we state the general security requirements of distributed applications, and identify a simple but crucial security requirement for the general acceptance of mobile code from the application developers' view. In Sections 4, we address the current status of mobile code paradigms in meeting these requirements. In Section 5, we present a travelling information retrieval application deploying the mobile agent approach, which is the most interesting branch of mobile code discussed, and study the security issues in this sample application. Finally in Section 6, we conclude with some directions for future work on mobile code development.

2. The Mobile Code Paradigm

2.1 Classification of Mobile Code Paradigms

The mobile code paradigm is actually a collective term, applicable wherever there is mobility of code. Different classes of code mobility can be identified, whereas Ghezzi and Vigna proposed three of them, namely *remote evaluation*, *code on demand* and *mobile agent* [GV97]. In what follows, we use the words "paradigm" and "paradigms" interchangeably, where the plural word "paradigms" refers to the three separate classes, while the singular word "paradigm" addresses the three classes as a whole. The classification, together with the client/server paradigm, is summarized in Figure 1.

<table>
<tr><th colspan="2">Paradigm</th><th>Local side</th><th>Remote side</th><th>Computation takes place at</th></tr>
<tr><td colspan="2" rowspan="3">Client/server</td><td rowspan="3">-</td><td>Know-how</td><td rowspan="3">Remote side</td></tr>
<tr><td>Processor</td></tr>
<tr><td>Resources</td></tr>
<tr><td rowspan="9">Mobile code</td><td rowspan="3">Remote evaluation</td><td>Know-how →</td><td></td><td rowspan="3">Remote side</td></tr>
<tr><td></td><td>Processor</td></tr>
<tr><td></td><td>Resources</td></tr>
<tr><td rowspan="3">Code on demand</td><td>← Know-how</td><td></td><td rowspan="3">Local side</td></tr>
<tr><td>Processor</td><td></td></tr>
<tr><td>Resources</td><td></td></tr>
<tr><td rowspan="3">Mobile agent</td><td>Know-how →</td><td></td><td rowspan="3">Remote side</td></tr>
<tr><td>Processor →</td><td></td></tr>
<tr><td></td><td>Resources</td></tr>
</table>

Figure 1. Ghezzi and Vigna's Classification of Mobile Code Paradigms.

In particular, the "know-how" in Figure 1 represents the code that is to be executed for the specific task. In the mobile code paradigms (*remote evaluation*, *code on demand*, and *mobile agent*), the *know-how* moves from one side to another side regarding where the computation takes place; while in the client/server paradigm, the *know-how* is stationary on the remote (server) side. *Resources* are the input and output for the code, whereas

processor is the abstract machine that carries out and holds the state of the computation. The arrows represent the directions in which the specific item should move before the required task is carried out.

Remote evaluation has the know-how on the local side, but the resources and processor on the remote side. The know-how is to be moved from the local side to the remote side for computation to carry out. The opposite happens for code on demand where the resources and processor reside on the local side, but not the know-how.

The mobile agent paradigm adds complexity to the remote evaluation paradigm by also allowing the processor to move around with the know-how. Therefore, agents can make different execution decisions according to different execution results on previous destinations. This idea is actually an extension of the intelligent agent concept applied in a distributed execution environment [RN95].

Ghezzi and Vigna's classification is found to be comprehensive and representative of most existing mobile code paradigms, and we will base our discussion on this classification.

2.2 Applications of Mobile Code Paradigms

While the mobile code paradigms are not yet common among application developers, they are already implicitly used in many systems. For example, the `rsh` command is utilizing the remote evaluation approach, and Java applets are realizing the code-on-demand approach.

The more complex mobile agent paradigm, however, is less commonly applied in distributed applications. Nevertheless, the enlightening idea of code autonomy in a de-centralized system has directed researchers to increasing efforts of making the paradigm realistic. [Whi98] suggested many mobile agent applications in everyday life. One pro-classical example in electronic commerce is autonomous bargaining and shopping. Other applications of the mobile agent approach are suggested in [PK98], [BGP97], [OOC97], and [BP98].

No matter which category of mobile code paradigms is employed in an application, users should always be specifically reminded of the underneath security risks. In Section 4, we will have a closer examination of security concerns for the mobile code paradigms.

2.3 Supporting Implementation Technologies

A paradigm (both mobile code and client/server) would be of little use if no implementation technology can support its realization. Fortunately, active research has been taking place to realize the client/server paradigm since the past few decades, as well as the mobile code paradigm in these few years [AS98].

Ghezzi and Vigna also gave a classification of the supporting implementation technologies in [GV97]. One branch of technologies they discussed is called *message-based* technology, which allows an executing unit to send messages to and receive messages from a remote executing unit. An example is the Remote Procedure Call (RPC). Another branch is called *mobile* technology: technologies that allow an executing unit to move their code to a remote side, possibly carrying their execution state when they move. Examples are the Aglets [IBM99], Mole [IPV98], and Odyssey [GM98]. In Section 5, we will present a mobile agent system implemented using Aglets.

3. A Security Requirement for Mobile Code

Before we go on to discuss the security concerns for mobile code paradigms, and the security features of mobile code technologies, we first state our scope on what we mean by security. It is common to discuss security in three aspects: *security attack*, *security mechanism*, and *security service* [Sta99]. We identify, in a typical process of application development, a central concern before the mobile code paradigm can be deployed and the mobile code technologies can be utilized:

MCS: An application developed using the mobile code paradigm can be as secure as the same application developed using the conventional client/server paradigm.

The identification of this concern is natural and straightforward. A typical application development process consists of the *requirement analysis*, *design*, *implementation*, and *testing* phases [Som92]. Given a set of security requirements in the requirement analysis phase, MCS concerns with the acceptance of the mobile code paradigm in the design phase of the application development process. Restating in the usual security terms, MCS is equivalent to saying that the mobile code paradigm must not bring additional security attacks that have no corresponding security mechanisms to counter with.

In Section 4, we explore possible security attacks to applications developed with the mobile code paradigm, and the security mechanisms that can account for these attacks. We then make a comparison with the client/server paradigm to see if one paradigm is more vulnerable to security attacks than the other. This would lead us to assessing the current status of the validity of MCS.

4. Security Concerns of Mobile Code Paradigms

In this section, we discuss some possible security attacks (Section 4.1) to different mobile code paradigms, and possible mechanisms (Section 4.2) against these attacks. To address MCS, we compare these attacks and mechanisms with the attacks and mechanisms corresponding to the client/server paradigm in Section 4.3.

4.1 Security Attacks

A security attack is an action that compromises the security requirements of an application. Applications developed using different paradigms are subject to different attacks, which would be discussed in the following subsections.

4.1.1 Security Attacks to the Client/Server Paradigm

Conventionally, the client/server paradigm assumes the security model of an "information fortress". In the information fortress model, the local computer is assumed to be a secure premise for code and data. Only the communication channels and remote sites would be suspected to be malicious. This effectively limits the source of security attacks to outsiders of the local machine, and eliminates the possibilities of attacks to the client on the client side and attacks to the server on the server side.

Therefore, the main possible attacks are *masquerading* (pretending the server or the client), e*avesdropping* on the communication channel, and *forging messages* to the client or the server. Consequently, the main challenges are to establish the appropriate trust relationship between the client and the server, and to secure the communication channel from being eavesdropped.

4.1.2 Security Attacks to the Mobile Code Paradigms

While the security fortress model is usually assumed in the client/server paradigm, it also applies to the remote evaluation and code-on-demand approaches, with some additional considerations. In the remote evaluation paradigm, the security fortress model applies with the additional concern that the remote side must make sure the code is not harmful to run. Therefore, apart from building trust between the sender and the receiver, an additional challenge is to verify the code in the defending mechanism. Furthermore, code-on-demand is similar to remote evaluation, as we see in Figure 1. The only difference between these two approaches is the switching of the roles between the remote side and the local side, and the challenge is also on code verification.

Mobile agent, on the other hand, is the most challenging area of mobile code security, due to the autonomy of agents. Mobile agent security is usually divided into two aspects: *host security* and *agent security*. Host

security deals with the protection of hosts against malicious agents or other hosts, while agent security deals with the protection of agents against malicious hosts or other agents.

For host security, the security fortress model can still apply, although agents may co-operate together to perform a complex attack [GBH98]. For agent security, however, a mobile agent can hardly be modeled as a "fortress", due to the lack of trusted hardware to anchor security with [Tsc99]. Consequently, here are two branches of possible attacks to agents:

i. *data tampering*: a host or another agent may modify the data or execution state being carried by an agent for malicious purpose
ii. *execution tampering*: a host may change the code executed by an agent, or rearrange the code execution sequence for malicious purpose

Until recently, it is generally believed that mobile agents are under the full control and mercy of hosts, thus agent security is an intractable problem. We will examine in the next section some recent development in agent security mechanisms that would likely break this limitation.

4.2 Security Mechanisms

Security mechanisms are mechanisms designed to prevent, detect or recover from security attacks. We examine the security mechanisms developed for different paradigms in this subsection.

4.2.1 Security Mechanisms against Attacks to the Client/Server Paradigm

We see from Section 4.1 that the main security challenges of the client/server paradigm are the mutual trust building between clients and servers, plus the protection of messages in transit. These problems can be satisfactorily solved by cryptographic techniques such as *security protocols* (e.g., Kerberos [MIT99] and the Secure Socket Layer version 3, SSLv3 [FKK96]) and *message encryption*. These mechanisms are already extensively employed in existing client/server applications. A lot of details can be found in [Sch96] and [Sta99].

4.2.2 Security Mechanisms against Attacks to the Mobile Code Paradigms

As possible attacks to mobile code paradigms are increasing, more mechanisms are required to secure mobile code applications. We see from Section 4.1 that the main additional challenge to security of mobile code paradigms is the verification of the received code. Two significant approaches to this problem are the *sandbox model* and the *verification techniques*.

In the sandbox model, the code or agent received from a remote side can only access a dedicated portion of system resources. Therefore, even if the received code or agent is malicious, damage would be confined to the resources dedicated to that code or agent. This is the main security mechanism of the Java programming language, and Java-based systems such as Aglets [KLO97]. In addition to sandboxes, received code or agents are filtered through a code verifier which checks integrity of the code, for example, no access to out-of-bound memory and type safety, etc. A host can also limit the period of time for a specific received program to run, and thereby reducing the risk of time-consuming attacks. Besides, code may be digitally signed for hosts to verify that they are actually from the intended senders.

Sandboxes and verification techniques serve to satisfy most of the host security requirements of mobile code applications. These techniques are well known and generally accepted. On the other hand, for agent security, no good mechanism is established to protect an agent from being tampered with. However, some approaches have been proposed, and they can be classified into two categories.

The first category is agent-tampering detection. These techniques aim at detecting whether an agent's execution or data have been tampered with along the journey. Some possible approaches are *range verification*, *timing information*, *addition of dummy data items and code*, and *cryptographic watermarks* [Tsc99].

Another category is agent-tampering prevention. These techniques aim at preventing agent code or data being tampered with. Two approaches are highlighted here:

i. *Execution of encrypted functions* [ST98]: with some specially designed functions, program can be implemented to evaluate encrypted functions, which cannot be understood by hosts, but still executable. The results of such evaluation would also be encrypted and trusted because no one would be able to forge the encrypted function, nor the encrypted result.

ii. *Time-limited black-boxes* [Hoh98a]: agents are messed up and obfuscated, such that they arrive to hosts as black boxes. Hosts would be able to evaluate the black boxes, but would not have a very good understanding of what the agent is meant to do. Besides, the lifetimes of black boxes are specified, making time-consuming attacks unlikely.

These approaches are enlightening in the way that they open new areas in computer security. Yet they provide limited protection to agents for the time being. Agent protection is still in its early stage, compared with the maturity of protection for hosts and client/servers, and efforts should be spent on improving the already-proposed mechanisms, or developing new protection mechanisms.

4.3 A Security Comparison Between Paradigms

From the previous discussion, we see that attacks to the client/server paradigm are least possible, due to the assumption of the security fortress model. Mechanisms are well established to defend these attacks. Remote evaluation and code on demand are subject to more attacks, but the mechanisms against these attacks are also quite well established. This is consistent with the popularity of Java applets, which is a typical example of code on demand. Mobile agents, on the other hand, are facing even more attacks, yet the defending mechanism is not very well established. Figure 2 summarizes the comparison. Therefore, from the view of an application developer, remote evaluation and code on demand are acceptable for software design; however, the acceptability of the mobile agent approach is still in question, i.e., MCS is not currently valid.

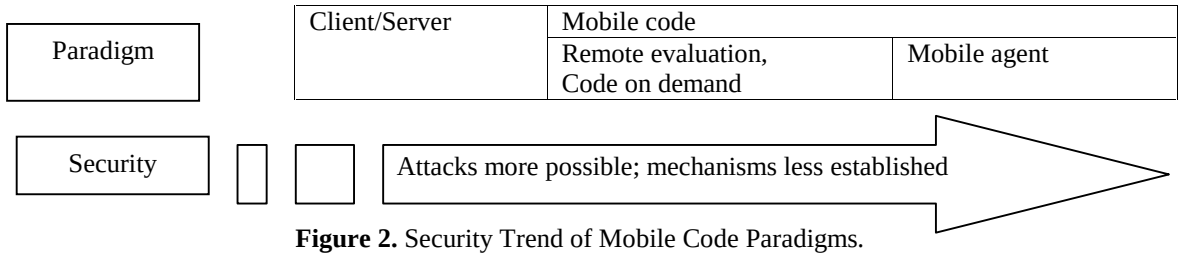

Figure 2. Security Trend of Mobile Code Paradigms.

5. A Case Study for Mobile Agent Security: the Traveling Information Agent System

In Section 4, we see that mobile agent security, especially agent protection, requires a particular attention. To study the security concerns of mobile agents in more details, we implemented a traveling information retrieval application, namely the *Traveling Information Agent (TIA)* system, deploying the mobile agent paradigm, and investigated possible attacks to the system we built. In this section, we describe the system briefly, and present some of the possible attacks the system is exposed to.

5.1 The System

The system can be divided into a client system and a server system. The client system is built on a handheld PC running Windows CE while the server system is built on a desktop computer. The server can be further divided into two parts: a database system (an Oracle server) and a proxy server that interfaces between the handheld PC and the databases. Users of the system inputs his request for information (such as transportation, accommodation, weather, news and exchange rate of a particular place) through a graphical user interface, the client system (on the handheld PC) analyzes the request, and sends an agent to the proxy server to ask for data. The proxy server gets the request from client, and relays the agent to the database servers. After arriving at the appropriate database, the agent gets the appropriate data from the server. Then, the agent returns to the proxy

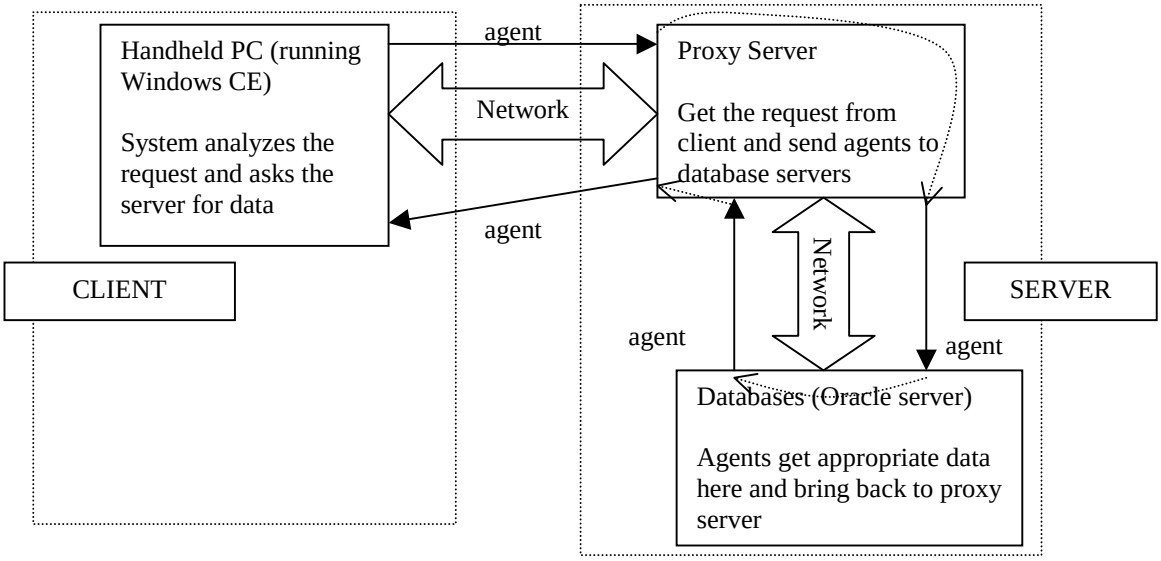

Figure 3. The Traveling Information Agent system.

server, and finally go back to the client. The system is illustrated in Figure 4.
Agents are implemented as Java objects using the IBM Aglets Software Development Kit [IBM99]. Notice that although we divide the system into the client part and the server part, it is actually a mobile agent application rather than a conventional client/server application. This is because we are sending and receiving agents between machines, there is mobility of both "know-how" and "processor", as described in Section 2.

5.2 Attack Model and Scenarios

As described in Section 4, there are two types of attacks to the mobile agent paradigm: attacks to hosts and attacks to agents. Attacks to hosts (such as code authentication and verification) are similar to attacks to the conventional client/servers, and can be well described using the sandbox model. We are more interested in the new area of attacks to agents.

To describe attacks to agents, we also need a model. We find little information regarding models for mobile agent system and security, especially concerning agent protection. The information fortress model is not applicable to model mobile agents, and the sandbox model is suitable for host protection only. [Hoh98b] is the only work that is suitable for modeling agent protection. In this paper, a model of attacks by malicious hosts against mobile agents is proposed. In the model, agents and (malicious) hosts are represented by abstract machines that execute the corresponding agent or attack programs. Abstract machines are modeled using RASPS (Random Access Stored Program plus Stack) machines. An agent RASPS depends on the host RASPS for accessing the environment and communicating with other agent RASPS's. The host RASPS has access to all of the host environment, and is able to read and manipulate the properties of all agent RASPS's running on that host, as well as control their execution. Figure 4 illustrates the model.

This model is plausible because it can be used to describe most malicious host attack scenarios including:

i. spy out and modify the whole data part of an agent
ii. spy out and modify the code part of an agent
iii. manipulate the code execution sequence of an agent
iv. manipulate the agent execution environment

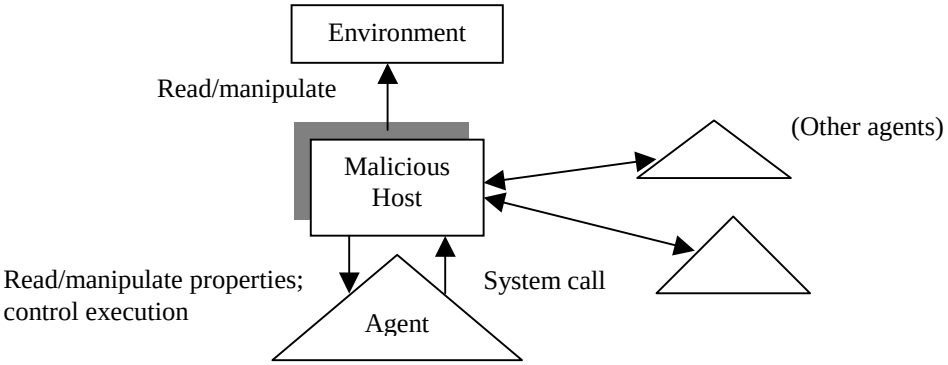

Figure 4. Attack Model of Malicious Hosts against Mobile Agents.

Applying the model to our TIA system, the agent is sent from the client handheld PC, the malicious host may be the agent servers running on the same host of the proxy server or the database server, and also other hosts that relay the agent. The environment may be the databases storing the traveling information (in case the database server is malicious). The following possible attacks to the agent can be identified:

i. a malicious host that relay the agent may spy out and modify data collected by the agent, therefore false information is reported to the user;
ii. a malicious host that relay the agent may spy out the code of the agent, thereby get to learn what information the particular user is interested in;
iii. a malicious host that is also running the database server may manipulate the execution sequence of or modify the code of the agent, and make the agent request some information from the database for it illegitimately;
iv. a malicious host running the database server may manipulate the information obtained from the databases, or modify the database directly, therefore the agent will obtain false information.

These are only a subset of all possible attacks to the agent in the TIA system. As described in Section 4. There is presently no good method to protect the agents from these attacks. Efforts should be spent on devising mechanisms to protect the agents.

6. Conclusion & Future Work

In this paper, we study the mobile code paradigm, which is a collection of remote evaluation, code on demand, and mobile agents, as an alternative to the conventional client/server paradigm. We examine security concerns of the mobile code paradigm, which is a bottleneck to the acceptance of mobile code from the application developers' point of view. We identify a crucial security requirement (MCS) of mobile code paradigms, and survey existing security attacks and mechanisms to evaluate the current status on meeting this requirement. We conclude that the mobile code paradigm is still to be developed in its security aspects.

From the study, we observe that mobile agent protection needs a particular attention. To investigate the security threats to mobile agents in particulars, we implemented a simple Traveling Information Agent system, and discuss the possible attacks to the agents in this system, based on the attack model in [Hoh98b]. In the

future, we would devote efforts to devising mechanisms that would enhance mobile agent security. We also want to implement and test these security mechanisms, and also evaluate these mechanisms. We believe that secure mobile agent applications would be a hot topic in the first century of the next millennium.

References

[AS98] The Agent Society. "Agent Code Available for Download". http://www.agent.org/pub/code.html, 1998.

[Bak97] Sean Baker. *CORBA Distributed Objects: Using Orbix.* Addison-Wesley, 1997.

[BGP97] M. Baldi, S. Gai, and G. Picco. "Exploiting code Mobility in Decentralized and Flexible Network Management". In Kurt Rothermet, Radu Popescu-Zeletin, editors, *Mobile Agents, First International Workshop, MA'97, Berlin, Germany, April 1997, Proceedings, LNCS 1219*, p. 13-26. Springer, 1997.

[BP98] L. Bernado, P. Pinto. "Scalable Service Deployment Using Mobile Agents". In Kurt Rothermel, Fritz Hohl, editors, *Mobile Agents, Second International Workshop, MA'98, Stuggart, Germany, September 1998, Proceedings, LNCS 1477*, p. 261-272. Springer, 1998.

[FKK96] Alan O. Freier, Philip Karlton, and Paul C. Kocher, "The SSL Protocol Version 3.0". Internet Draft. http://home.netscape.com/eng/ssl3/draft302.txt, November 18, 1996.

[GBH98] Michael S. Greenberg, Jennifer C. Byington, and David G. Harper. "Mobile Agents and Security". In *Volume 367, IEEE Communications Magazine*. IEEE Press, July 1998.

[GM98] General Magic Inc.. "Agent Technology". http://www.genmagic.com/technology/mobile_agent.html, 1998.

[GV97] Carlo Ghezzi and Giovanni Vigna. "Mobile Code Paradigms and Technologies: A Case Study". In Kurt Rothermet, Radu Popescu-Zeletin, editors, *Mobile Agents, First International Workshop, MA'97, Berlin, Germany, April 1997, Proceedings, LNCS 1219*, p. 39-49. Springer, 1997.

[Hoh98a] Fritz Hohl. "Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts". In Giovanni Vigna, editor, *Mobile Agents and Security, LNCS 1419*, p. 92-113. Springer, 1998.

[Hoh98b] Fritz Hohl. "A Model of Attacks of Malicious Hosts Against Mobile Agents". In *Fourth Workshop on Mobile Object Systems (MOS'98): Secure Internet Mobile Computations*, http://cuiwww.unige.ch/~ecoopws/ws98/papers/hohl.ps, 1998.

[IBM99] IBM Tokyo Research Laboratory. "IBM Aglets Software Development Kit Homepage". http://www.trl.ibm.co.jp/aglets/index.html/, 1999.

[Inp98] Inprise Inc.. "Visibroker, CORBA Technology from Inprise". http://www.borland.com/visibroker/, 1998

[IPV98] IPVR, University of Stuggart. 'The Home of Mole". http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole.html, 1999

[KLO97] Günter Karjoth, Danny B. Lange, and Mitsuru Oshima. "A Security Model for Aglets". In Giovanni Vigna, editor, *Mobile Agents and Security, LNCS 1419*, p. 188-205. Springer, 1998.

[Lin97] J. Linn. "Generic Security Service Application Program Interface, Version 2". RFC2078 (Obsoletes: RFC1508), 1997.

[MBB+98] Dejan Milojicic, Markus Breugst, Ingo Busse, John Campbell, Stefan Covaci, Barry Friedman, Kazuya Kosaka, Danny Lange, Kouichi Ono, Mitsuru Oshima, Cynthia Tham, Sankar Virdhagriswaran, and Jim White. "MASIF: The OMG Mobile Agent System Interooperability Facility". In Kurt Rothermel, Fritz Hohl, editors, *Mobile Agents, Second International Workshop, MA'98, Stuggart, Germany, September 1998, Proceedings, LNCS 1477*, p. 50-67. Springer, 1998.

[MIT99] Massachusetts Institute of Technology. "Kerberos: The Network Authentication Protocol". http://web.mit.edu/kerberos/www/, 1999.

[OMG98] Object Management Group. "CORBA Security Service Specification (1.2)". http://www.omg.org/corba/sectrans.html#sec, December 1998.

[OOC97] L. A. G. Oliveira, P. C. Oliveira, and E. Cardozo. "An Agent-based Approach for Quality of Service Netgotiation and Management in Distributed Multimedia Systems". In Kurt Rothermel, Radu Popescu-Zeletin, editors, *Mobile Agents, First International Workshop, MA'97, Berlin, Germany, April 1997, Proceedings, LNCS 1219*, p. 1-12. Springer, 1997.

[PK98] Vu Anh Pham and Ahmed Karmouch, "Mobile Software Agents: An Overview". In *Volume 367, IEEE Communications Magazine*. IEEE Press, July 1998.

[RN95] Stuart Russell and Peter Norvig. "Intelligent Agents". In *Artificial Intelligence, A Modern Approach*, p.31-50. Prentice Hall, 1995.

[Sch96] Bruce Schneier. *Applied Cryptography*. Wiley, 2nd edition, 1996.

[Som92] Ian Sommerville. *Software Engineering*, 4th edition. Addison-Wesley, 1992

[ST98] Tomas Sander and Christian F. Tschudin. "Protecting Mobile Agents Against Malicious Hosts". In Giovanni Vigna, editor, *Mobile Agents and Security, LNCS 1419*, p. 44-60. Springer, 1998.

[Sta99] William Stallings. *Cryptography and Network Security, Principles and Practice*. Prentice Hall, 2nd edition, 1999.

[Tsc99] Christian F. Tschudin. "Mobile Agent Security". In M. Klusch, *Intelligent Information Agents*. Forthcoming *LNCS*. http://www.docs.uu.se/~tschudin/pub/cft-1999-iia.ps.gz, 1999.

[Whi98] Jim White. "Mobile Agents White Paper". http://www.genmagic.com/technology/techwhitepaper.html, 1998.