SECURITY OF MOBILE AGENTS

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF TECHNOLOGY IN INFORMATION TECHNOLOGY

(SOFTWARE ENGINEERING)

Under the Supervision of
Dr.Shirshu Verma
IIIT-Allahabad

Submitted By
Prabhat Singh S.
M. Tech. IT (SE)
MS200515
IIIT-Allahabad

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD

(A University Established under sec.3 of UGC Act, 1956 vide Notification No. F.9-4/99-U.3 Dated 04.08.2000 of the Govt. of India)

(A Centre of Excellence in Information Technology Established by Govt. of India)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY ALLAHABAD

(A University Established under sec.3 of UGC Act, 1956 vide Notification No. F.9-4/99-U.3 Dated 04.08.2000 of the Govt. of India)

(A Centre of Excellence in Information Technology Established by Govt. of India)

Date: _____

WE DO HEREBY RECOMMEND THAT THE THESIS WORK PREPARED UNDER OUR SUPERVISION BY *PRABHAT SINGH S.* ENTITLED *SECURITY OF MOBILE AGENTS* BE ACCEPTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF TECHNOLOGY IN INFORMATION TECHNOLOGY (SOFTWARE ENGINEERING) FOR EXAMINATION.

COUNTERSIGNED

Dr. SHIRSHU VERMA
(THESIS ADVISOR)

Prof. U. S. Tiwary

DEAN (ACADEMIC)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY ALLAHABAD

(A University Established under sec.3 of UGC Act, 1956 vide Notification No. F.9-4/99-U.3 Dated 04.08.2000 of the Govt. of India)

(A Centre of Excellence in Information Technology Established by Govt. of India)

CERTIFICATE OF APPROVAL*

The foregoing thesis is hereby approved as a creditable study in the area of information technology carried out and presented in a manner satisfactory to warrant its acceptance as a pre-requisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it is submitted.

COMMITTEE ON FINAL EXAMINATION FOR EVALUATION OF THE THESIS	_____ _____ _____

* Only in case the recommendation is concurred in

CANDIDATE DECLARATION

This is to certify that Report entitled “Security of Mobile Agents” which is submitted by me in partial fulfillment of the requirement for the completion of M.Tech. in Information Technology (with the specialization in *Software Engineering*) to Indian Institute of Information Technology, Allahabad comprises only my original work and due acknowledgement has been made in the text to all other material used.

Prabhat Singh S.
Roll No. : MS200515

ABSTRACT

In the past few years the computer systems have evolved from monolithic computing device to much complex client-server environment. Now new phase of evaluation allows complete mobility of application code among supporting platforms to form a loosely-coupled distributed system. Mobile-agent paradigm is one such technology. It has numerous application where it can be beneficial, to name a few areas where the mobile-agents have potential deployment are database search, distributed systems and e-commerce. This technology has given a new direction to networking. It can produce very good results in limited resources or in deficient environment where bandwidth, memory are significant constraint. But still it is not widely accepted due to its security issues.

The title of this thesis "*Security of Mobile Agent*" suggests itself about the work emphasized in this thesis. The main area of discussion is the security of mobile agents on malicious host. Here a malicious host refers to a system in a network which can take advantage of the vulnerabilities of a mobile agent that has come to the host machine to get its work done. This thesis discusses about the mobile-agent, the various security threats that can be posed by malicious host and consequently the solutions. This thesis proposed a solution by combining few solutions and distilling the best from the solutions so that it can provide a better solution. Finally this thesis concludes with implementing the solutions and the results obtained by the experiments using trading example.

ACKNOWLEDGEMENT

It was a pleasure doing the thesis work which helped me learn new things, coming out with solutions to the toughest problems and develop my programming skills. First and foremost I would like to thank **INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD** for providing me such opportunity to carry out the dissertation work.

I would like to express my sincere gratitude to my thesis supervisor **Dr. Shirshu Verma** for providing his precious advices and suggestions. I would like to thank him for spending his valuable time in correcting my mistakes and posing new challenges which really made me work hard on it. His guidance, both in terms of technical advice on my research and in terms of professional advice, was invaluable. He always raised the bar inspiring me to strive for the best, and was always among the first ones to applaud any accomplishment. With his valuable remarks the work was finally completed.

I would also like to thank my class mates who helped me out time to time. **Mr. Kamal Sawan** for helping me out in the testing and debugging process The remaining names are**, Mr. Imran Khan, Mr. Anand Arun Atre, Mr. Vineet Chauhan, Mr. Nilesh Shukla, Mr. Abhay Pawane, Mr. Pankaj Kandpal, Mr. Sahab Nath Yadav, Mr. Anil Pandey, Ms. Poonam Yadav and Mr. Sampath Kumar.** They were quite patient in doing so and also contributing their valuable time for me in completing the work.

Finally, I would like to thank my parents **Sh. Jitender Kumar** and **Smt.Santosh**, my brother **Mr. Yogesh Singh S.** and sister **Ms. Pratigya Singh S.**, who always supported me and encouraged me for higher studies. I would not be writing this thesis today without their constant support and devotion. I dedicate this thesis to them.

Prabhat Singh S.

Detailed table of contents

List of Figures

List of Tables

Chapter 1
Introduction - Objective & Scope

This chapter introduces the objective and scope of this thesis. This chapter tells who should read this thesis and what to expect from this thesis. This thesis concludes with the organization of the thesis.

1.1 Overview

This thesis focuses on the security issues of a mobile agent in an e-commerce domain, where agents are capable of buying and selling items in an open market and is open to all kind of attacks like alteration, manipulation, eavesdropping etc. this agent is vulnerable to many attacks as it is executing on some other host and that host may take advantage of this authority. This work shows the possible countermeasures taken to avoid the well-known attacks caused by a malicious host.

1.2 Objective

The mobile agent paradigm is applicable in many application fields from which some areas are specially recognized as e-commerce, electronic auctions and stock market. These areas need high transaction of money. Now the user will put his money in mobile-agent only when he is confident that his money is secure and the agents can be trusted

which are dealing with his money or transmitting some secure information. This is one reason which instigates much research effort in security of mobile-agents which has its own benefits when used in these areas.

As the name suggests that these agents are mobile in nature, which make them to travel autonomously in the network, due to this nature the agents become more vulnerable to various attacks. These attacks expose the limitation of mobile agents in the area of security. Until all or most of these attacks are resolved the usage of mobile-agents to its full potential will remain a constraint.

The objective of this thesis is to recognize all countermeasures taken to avoid the attacks caused by a malicious host. These attacks may include the stealing of private data like credit-card number or e-money which it uses to buy the items, or manipulation in the results of an agent that it generates on the remote host and finally to protect itself from manipulation or alteration of its code itself. This work shows the use of cryptography methods to solve the basic attacks and to protect the agent from the bigger attacks it uses the combination of encryption and obfuscation techniques to solve the problem.

1.3 Target Audience

This thesis deals with the security, agent system and computer language JAVA. So it will be easier for the reader if s/he has the background knowledge of following technology:

1. Cryptography techniques.
2. Agent technology.
3. Any mobile agent system based on JAVA.

1.4 Organization of Report

This report is organized in to six chapters. The first chapter gives the introduction of the thesis and tells the objective of the work, it tells the reason and motivation behind this work. Chapter 2 introduces the agent and mobile-agent technology, it discusses about the architecture of mobile-agent and gives the difference between the other technologies and mobile-agent paradigm. Chapter 3 discusses about the agent platform which is used to develop the agents and a brief discussion about the other agent system which are currently available. Chapter 4 discusses the security issues related to mobile agents and elaborate the proposed countermeasures. Chapter 5 shows the results obtained by providing the snapshots. Finally Chapter 6 summarizes the work done by giving the conclusion and the possible future work.

Chapter 2
Mobile Agent Technology

This chapter introduces the Agents, Mobile Agent and their advantages on traditional system being used. After reading this chapter you will have thorough understanding of agent technology and the advantages of the same over other existing technologies.

2.1 Agents

Most obvious definition of an Agent in real world is something who acts on the behalf of somebody. They are given some goals and they try to achieve these goals according to their intelligence. The basic dictionary definition of agent is *one who acts* [1]. In order to achieve these goals they communicate and interact with other agents, they exchange information and take back the results to the user. The software agents behave in the same manner too. Hyacinth S. Nwana says in his paper that agents may be stationary, may be resident or mobile able to move from one host to another [2]. The necessary points for the IT system agent are they must be software, hardware or a combination of those. Developers have found various forms of agents in IT system. Few of these are discussed below:

2.1.1 Software Agents

These are the agents that are implemented using software. This mean they are autonomous and they can react with other entities like other software agents, machines and humans on various platform. All other agents are some or other form of these software agents. Generally the term agent will mean software agent.

2.1.2 Autonomous Agents

Autonomous agents are those agents who react to their environment. They interpret the conditions of its environment and reacts according to that. In doing so, it can take decisions on its own. This is the important property observed by FIPA [3]. Agents are dynamic entity that means they can take decisions on their own to some extent, if there is no autonomy the agents can no longer possess dynamic behavior.

A proactive agent exhibits higher degree of autonomy where they depend not only on the input taken from the environment but also depends on the internal state of the agent. This means that they use memory for this purpose by taking help of neural networks. James Odell says in his work that proactive agents will actually poll the environment for events and other messages to determine what action they should take [4].

This decision taking ability of the agents may lead to the nondeterministic behavior of the agent. A shopping agent's may be highly unpredictable as it may show up with some good results or no results at all.

2.1.3 Intelligent Agents

The intelligent agents are those which have the ability to learn and adapt according to the situations. They use artificial intelligence to achieve these abilities. Intelligent agents are able to choose one situation out of given number of situations to go one step near to its goal. Its every action takes it near to its goal.

2.1.4 Adaptive Agents

An agent is an agent only in its environment, so if the environment is changed, you may or may not have the agent. For example a robot with the vision sensors in a room is an agent but when the lights are turned off the robot may not be an agent any more. If an agent can cope with the changing environment it is known as adaptive agents. These adaptive agents can adapt themselves to the changing environment. To exhibit this sort of ability the agent use neural networks, Bayesian rules etc. example of adaptive agents can be thermostat, simple search bots or robotic sensors.

2.1.5 Mobile Agents

Danny B.Lange and Mitsuru Oshima in its book says a mobile agent can be thought of as a software program which travels from one platform to another in order to get its work done, during this process it carries its state and data with itself and resume its execution from the state it had left on the previous platform [5]. The reason for using mobility is the improved performance which can be achieved by moving the agent closer to the new host, where it can use the services locally. We can take an example where agent needs information from several host situated on different platforms. It can use remote procedure call (RPC) where it can request the desired information and obtain the results by invoking the remote methods. This RPC follows the client-server paradigm. But if the volume of data is large it can create bandwidth and network traffic problem. In such cases the mobile agent can migrate to those remote hosts and perform the functions locally and come back with the desired results. It would be a more efficient way to process the data.
The ability of an agent to migrate from one environment to another is not a requirement for agent hood. Still mobility is an important property for many agent-based systems and necessary for a certain class of application.

2.1.6 Coordinate Agents

Coordinate agent are those agents which work with some degree of coordination among themselves. As humans work in coordination, the agents belonging to different systems can use the same pattern. Some application using coordinate agents are supply chains, scheduling, problem solving, contract negotiation etc. these application need some degree of coordination among themselves otherwise these kind of systems are not possible. When we talk about coordination we take heterogeneous population of agents so we need extra care while designing these kind of agent system as some concepts like how will they negotiate, manage, cooperate, compete etc will come into picture.

2.1.7 Other forms of agents

The agents mentioned above are predominate forms of every agent based system. More detail work on the applications can identify other forms of agents like broker agent, manager agent, facilitator agent etc.

2.2 Agents and Object Technologies

Agent and the object technologies both have their benefits, so both the technologies are equally fascinate the developers. Object and agents are very similar to one another, objects does most of the work which agents also do. So there should be some additional benefits of using the agents over objects. We will look at those differences here.
Below are few points which tell the differences and similarities between agents and objects[1]:

Similarities:

Objects and agents are alike in several ways. James Odell explains in his work on agents that we can assume agents as the objects or components or entities as they have their own identity, have their own state and behavior, which distinguish agents from one another. They have interface through which they can communicate with each other [4].

Differences:

Agents are autonomous and reactive - one of the most important properties of the agents is that they are autonomous in nature. So they can take their decisions and can decide which message to respond. This is the property which objects don't possess, they only do what they are asked to do. Agents are also reactive as they are always listening and react to the situation.

Intelligent Functionality – agents are always considered to be more functional and intelligent than the single object. Agents also indulge in multi-agent system activity, where agents interact and hence found to be more functional than the objects. These agents exhibit complex reasoning for which they use inference engines or neural networks.

Agent Communication Language – to communicate, the agents use a powerful language to communicate to each other. They are able to represent the complex desire-belief-intention information. After observing the results, they can draw inferences and can even manipulate the behavior or functionality. On the other side objects uses the fixed set of messages while communicating.

These are the few basic differences which differentiate an agent to an object, which shows the benefits of agents over the objects. There are more benefits which show that the agents are much more useful than the objects.

2.3 Status of Agent Technology

Agent technology is not a single technology, it's a integrated form of multiple technologies. Agent technology adds the additional capabilities or functionality to the existing technology not actually providing the new set of capabilities.

The current state of agent technology is as follows[1]:

- It is still an active research area.
- The full set of technologies is not available.
- The technologies are not integrated with one another.
- Agent technology is still not a very widespread technology or has been widely accepted as it seems from its benefits.

2.3.1 Usage of Agents

As already discussed, the agent technology is still growing. The usage of agent technology are few and but they are on increase. Some usages are discussed below[1]:

- **User assistance agents:** these agents are used at the user level where they provide services to the user. These types of agents offer information or give advice to the user. Big companies like Microsoft, Apple and Lotus have shown their interest in this area. The best example of these agents in common use is the animated help characters used in Microsoft Office products. These agents use Bayesian networks to analyze and to predict the possible topic where user may be needing help.
- **Organizational structure agents:** organizational structure agents imitate the human organization in the way they work. We can take an example of multi-agent supply chain systems where agents performs various roles like buyer, suppliers, brokers, manufacturing cell etc. these agents work in the form of organization and uses the complex algorithms so as to perform negotiation and other communication methods.
- **Network and system management agents:** The telecommunication companies are most active in this area. Colin G. Harrison, David M. Chess, and Aaron Kershenbaum shown in their technical report that the agents are to assist in complex system and network management tasks, such as load balancing, failure anticipation, problem analysis and information synthesis [6].

- **Interest matching agents:** these are the agents which has good potential in the marketing business. These agents suggest the recommendation such as “if you have bought this item, you would also like to buy this item”. These types of

matching interest agents are useful in the commercial web sites. These agents observe the interest pattern and give recommendation to the user. They have also already been deployed in various CD and video sales sites.

These agents represent some possible aspects to the usage of the agents, but still no-one shows the complete features of the mobile agents.

2.4 Mobile Agents

I would like to start this topic with a statement of Benjamin Grosof who has referred to this as the "software engineering" argument:

"Each individual advantages provided by mobile agents can be addressed more effectively in some ad hoc manner, but no technology addresses the advantages provided by mobile agents all at once."

A mobile agent can be thought of as a software program which travels from one platform to another in order to get its work done, during this process it carries its state and data with itself and resume its execution from the state it had left on the previous platform.
The importance of doing this is to improve the performance, as the agent is moved nearer to the host and it can use the services available on the remote host locally.
We can take an example of mobile agent application and discuss the issues related to it.

We can take an example of the agent application in a database search. In this example there are users who want to access the database situated at the various remote locations. The users need some information as the price quotation from the remote databases, now being at the remote location they cannot access the database directly and even so there may be the problems of the communication links as they cannot give the hundred percentage availability. How this problem can be rectified using the mobile agents.

One solution can be, to use the mobile-agent system, for this an agent server has to be installed on every host. The remote hosts may be laptop or palmtop machine with either dialup or wireless communication devices.

The software need to install on every host will consists of an agent runtime environment.

After setting the proper credential and permission on account of the agents, the mobile agents can be created which can move to the remote host to execute. As the agent's application is much easier to create they can be put in the production. And the work to search the database to retrieve the price quotation can be efficiently handled by the agents.

Danny B.Lange and Mitsuru Oshima shown the advantages of Mobile Agent programming [5]:

- They facilitate high quality, high performance, and economical mobile application.

  As the mobile agents reach to the remote system to execute, they exhibit high performance as the agents are executing the resources locally. So instead of fetching the data remotely, they process the data locally after reaching there. This exhibits the higher performance operation.

- They efficiently and economically use low bandwidth, high latency, error prone communication channels.

  As the agents are processing the data locally after reaching the remote host, it uses less bandwidth as the bandwidth is saved from the alleviating unnecessary remote calls. It efficiently works on error prone communication system.

- Ability to operate asynchronously and autonomously.

  Agents work asynchronously and autonomously so they move themselves to the location where they can find the services they need to perform the execution.

They can take the decisions on their own and can migrate in the network to effectively execute the tasks given to it.

- They are naturally heterogeneous.

  Network computing is basically heterogeneous, from both the perspectives hardware as well as software. Agent can work in any system where the agent platform is installed. So any hardware can support the execution of agents where the agent platform can be installed.
- They are robust and fault-tolerant.

  As the agents can take decisions and can react dynamically, it helps making the system more robust and fault-tolerant, where the agents can move to another host in the case of the failure of the current host.

2.5 Mobile Agent and Traditional System

Some advantages are shown when we compare the mobile agent to the client server technology.

Mobile agents take the client-server paradigm to the next level. Mobile agent paradigm gives the abstraction to the networking as with the help of mobile agents we don't have to worry about the underlying protocols working, how the communication takes places between agents and the migration of the agents over the network. All these issues are tackled by the agent platform itself.

The use of mobile agents greatly decreases the problem of robust networks. It also helps in upgrading the servers or moving the services from one system to another or implying the security services on the individual system without breaking the clients or revision of the network.

2.6 Architecture of Mobile Agent

The architecture gives the structure of the system which consists of some components, their individual functionalities and their interrelationship with each other. The basic architecture of the mobile agent can be thought of as a client sends out an agent which travel the network visiting servers in order to perform some required action.
The architecture consists of [5]:

- **Agent Manager:** agent manger has few responsibilities as it
  - Sends agents to and receives agents from remote hosts.
  - Prepares agents for transport by serializing the agent.
  - Reconstructs received agents and creates the agents execution context.

- **Security Manager:** responsibilities of security manager are:
  - Authenticates agents before allowing execution.
  - Automatically invoked when the agents tries to use any system resource or tries for any unauthorized activity.
  - Protects the host and agent from unauthorized access.

- **Inter-agent Communication:** it allows the agents to communicate through massage passing mechanism. This part is still under research work where agents from different agent systems can communicate to each other. Till now all those agent system which follow FIPA (Foundation for Intelligent and Physical Agent) standards [3] are able to exchange messages as they follow a standard format for sending and receiving messages. Inter communication is still an issue among heterogeneous agent system.
- **Directory Manager:** Lists names and addresses of services and agents. As shown in Fig.2.1 shows the agent first migrates to remote container and registers itself to the Directory Services. When any other agent needs to find some agent it contacts the Directory Services for help.

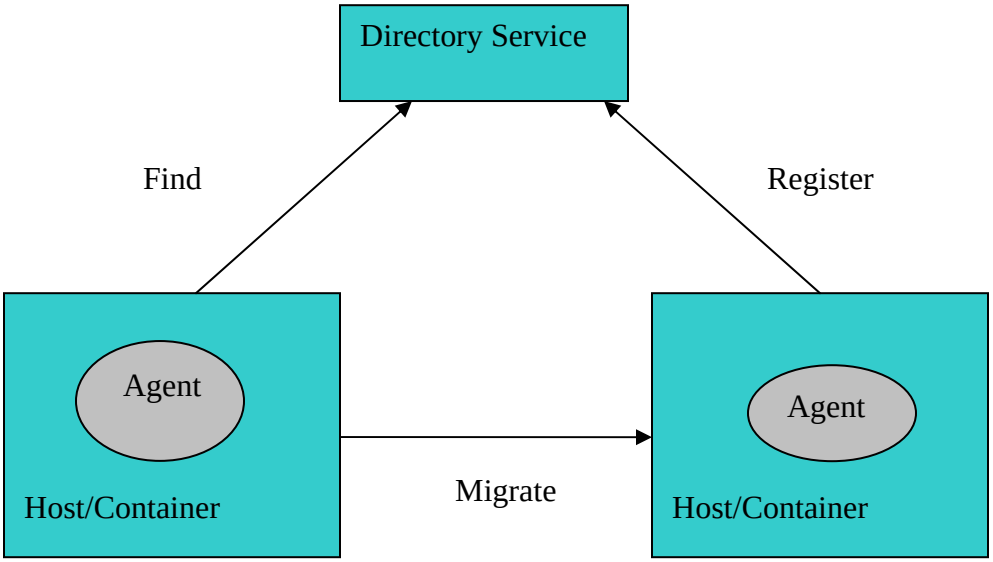

Figure 2.1

- **Language:** the architecture of a mobile agent system describes the flow of information among the various components of the system. The language used for the efficient transfer of data and provides the developer with the tools to efficiently implements the system. Most current agent systems are implemented on top of the Java Virtual Machine (JVM), which provides object serialization and basic mechanism to implement weak mobility.

2.7 Mobile Agent application

As our discussion on Mobile Agents it seems like it is a revolutionary technology but still it is not widely used. This is mainly because of its security issues, if all the

security requirements are addressed properly it will become a widely used technology. All the work which is being done by traditional system can be effectively done by agent technology as we have already discussed.

Apart from the usages discussed in **section 2.3.1** there are other applications where the mobile agents are very useful, these are discussed below:

- Resource availability, discovery, monitoring: agents are very useful in monitoring distributed data sources. They are useful in such application as they don't amplify network load, they can check the resource availability on network, they can monitor the system load etc. also the agents can be dispatched to the location where the user has no normal access to.
- Information Management Tasks: information management tasks like searching for information, information filtering, information monitoring. Search agents contain domain knowledge about various information sources. This knowledge includes the types of information available at each source, search agents are useful as they do their work of searching the relevant information from each source. It also deals with the problem of information overload to a user by limiting the information coming to a user.
- Dynamic software deployment: the deployment and installation of any application in a distributed environment is a complex task, particularly for unmanaged nodes in large scale deployment scenarios. Mobile agents can do this job more effectively as they do not require particular user intervention and its autonomous behavior helps in such situation very well.
- Personal agents
    - Email and news filters.
    - Personal schedule management.
    - Personal automatic secretary.
- Business-to-business applications
    - Market making for goods and services.

- Brokering of the above.
- Team management.

2.8 Available mobile agent system

Most of the agent systems are developed using:

- Programming language: Java or C++
- Communication language: KQML or FIPA ACL
- Java serialization is used for the migration of the agents.

The following Table 2.1 shows few of the mobile agent system available [1]:

Product	**Company**	**Language**
Agent Building Environment	IBM	C++, Java
Aglets	IBM Japan	Java
Concordia	Mitsubishi Electric	Java
Grasshopper	IKV++	Java
JADE	Tilab	Java
Microsoft Agent	Microsoft Corporation	Active X
Voyager	Object Space	Java

Table 2.1

2.9 Summary

This chapter discusses thoroughly about agent and agent system. After that it tells about mobile agents and its differences with traditional client-server system. With an example where it uses an agent for database search operation, it discusses the advantages of the same over traditional client-server system. It also tells where the agents are being used and components of its architecture. Finally it shows all the agent system which were in the market, out of which some became obsolete and some are still in use for research and for developing general application. In the next chapter we will be discussing about the various agent systems mainly Aglets on which we have developed our agent.

Chapter 3
Agent Platform

This chapter introduces the aglet platform. In this chapter we will briefly discuss about the Aglet System. Our main focus will be on discussing the inbuilt security options provided by the Aglets system. Then we will be discussing about the benefits and drawbacks of Java programming language used to write the agents and the security options provided by it. Finally we will see other java-based mobile agent systems.

3.1 Aglets

Mobile agents are the basis of the emerging technology which makes it easier to design, implement and maintain distributed systems. Mobile agents have the unique ability to transport themselves from one system in a network to another. We found that mobile agents reduce network traffic, overcome network latency and most importantly their ability to operate asynchronously and autonomously. It helps us to construct more robust and fault tolerant systems.

Aglets are Java objects that can move from one host on the internet to another. That is, an aglet can execute on one host, can stop its execution, dispatches itself to other host and

resume its execution on new remote host. On moving from one system to another aglet carries its code and data with it. The word aglet means “lightweight agent” in much the same way that applet means lightweight application. The term aglet is a combined work of agent and appletAglet is developed by research team at the IBM Tokyo Research Laboratory in Japan in early 1995 and is now open source [7].

Aglets are hosted by an aglet server in a similar way in which an applet is hosted by a web browser. The Aglet server provides an environment where agents can execute and Java language and Aglet security manager make the agents transfer safe. The Aglets Software Development Kit (ASDK) is an implementation of an Aglet API. The ASDK includes Aglet API packages, documentation, sample aglets, and the Tahiti Server. Tahiti is java application that allows the users to receive, manage, and send aglets to other computers that are running Tahiti.

3.1.1 Mobile Java Agent: The Aglet Model

The aglet model allows the proper and easier use of the agents created. The aglet model helps in setting up proper infrastructure through which we can use and take advantage of the agents. Here later in this chapter we will see the basic elements of aglet system briefly we will not go in the details. Danny B.Lange and Mitsuru Oshima has given the details of aglets and aglet model [5].

3.1.2 Basic Elements

The aglet object model explains some abstraction and the behavior which is used to take full advantage of this agent technology. The abstractions which are used are:

- Aglet: an aglet is a java object which moves in a network and gets executes on host which are aglet enabled. It is autonomous and run in its own thread. Proxy: a proxy is a representative of an aglet. It also protects the aglet from direct access to its public methods. The proxy also provides the location transparency for the aglet.
- Context: a context is where an aglet executes. It is a stationary object that provides a means for maintaining and managing aglets.

- Identifier: an identifier is bound to an aglet. This identifier is globally unique an immutable throughout the lifetime of the aglet.

The following list summarized the fundamental operation of an aglet as is shown in Fig. 3.1:

- Creation: the creation of aglets takes place in a Context. The new aglet is assigned an identifier, inserted into the context, and initialized.
- Cloning: the cloning produces exact copy of the original aglet in the same context. The cloned aglet has different identifier.
- Dispatching: dispatching an aglet from one context to another will remove it from its current context and insert into the destination context, where it will restart execution. This process is termed as dispatching.
- Retraction: the retraction will pull aglet from its current context and insert it into the context from which the retraction was requested.
- Activation and Deactivation: the deactivation of an aglet will halt its execution for the mentioned amount of time and store its state in secondary storage. Activation will again restore it in the same context.
- Disposal: the disposal of an aglet will halt its current execution and remove it from its current context.

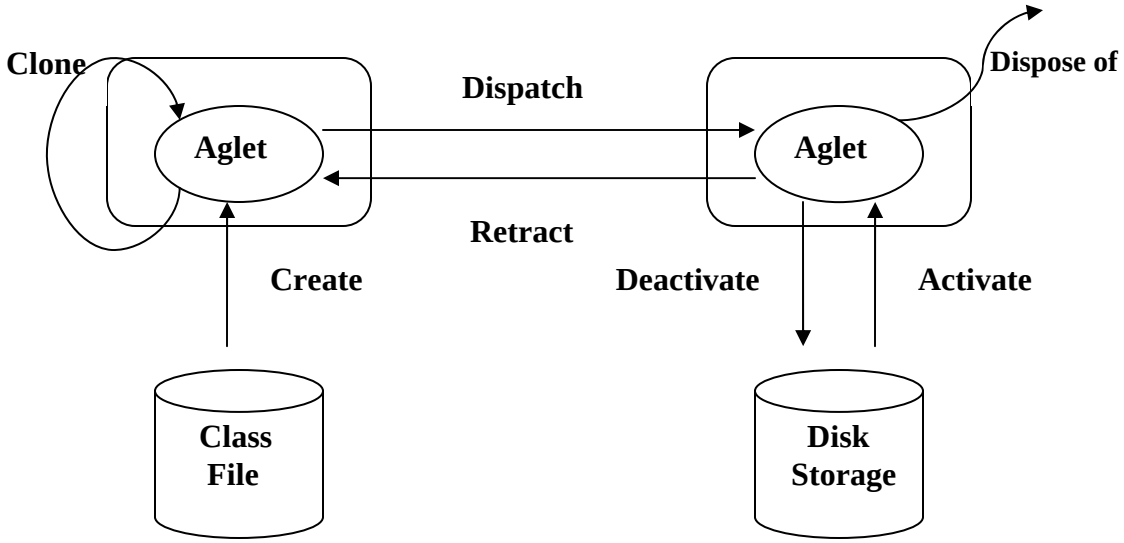

Fig 3.1

3.2 Security Services

Security services are important for the protection of your agents and server from the attacks. The following is the list of commonly available services for securing agent system [5].

- Authentication: before accepting an incoming agent, you want to know who its sender is. In this case, you need authenticate the agent. This process includes the verification of the developer who created the agent or before sending the agent to some host you may wish to know who the host is and what its credentials are.
    - Authentication of user: the user needs to authenticate himself to a given server. Public-key encryption or a password can be used for this purpose.
    - Authentication of host: before a server starts to communicate with another server or client, it needs to know with whom it is communicating.
    - Authentication of code: before executing an incoming agent, the host needs to know who created the agent. Digital signatures are typically used for this purpose.

- Authentication of agent: before executing an incoming agent, the server needs to know who is responsible for this agent or who its owner is.

- Integrity: to trust an agent, you need to make sure that no one has tempered with its code and data. Checking the integrity of the agent is the technique we use to make sure that no manipulation is done with its code and data. Confidentiality: an agent may carry confidential information that should be readable only by intended server or agent. Such information should be kept secret form other servers and agents.
- Authorization: authorization or access control is the way to specify and enforce an agent's capability to access information or to use services provided by a server.
- Nonrepudiation: an agent or sever cannot deny that a given communication exchange or transaction has taken place.
- Auditing: auditing service records security-related activities of an agent for later inspection.

3.3 When you cannot protect your Agents

The protection of your agents depends mostly on the server on which it is sent. Some of the points are shown where an agent is difficult to protect [5].

The first threat is encountered when the agent is migrated to another host over a network. Malicious host party may try to eavesdrop that may lead to the intercepting the agent's communication or try to retrieve the agent's private information. The third party may also try to tamper with the content of the agent that may lead to the unwanted behavior of the agent. This kind of attack is easy to handle and can be solved by using secure connection, such as Secure Socket Layer (SSL), which can encrypt the data and check their integrity.

- The next threat would be from the server where the agent is going to execute. When you send an agent to the remote site, you can never be sure that the host will treat your agent properly. As an agent owner you need to consider the following issues:

- Agent Execution: as the agent is executing on remote host, the agent is completely under the mercy of the host. The remote host may or may not be malicious and if it is malicious one then it can feed wrong information to the agent or may misled the agent to next host. So we need to take additional countermeasures before sending the agent to some host or only to some trusted host. Information that is secret to servers: as the agent is executing on the remote server, the server has the potential to steal, tamper with, or leak any of the agent's information. Here an agent cannot generate the secret information without the consent of the server. This implies that agent cannot keep secret communication with other agent from the server.
- Information secret from other Agent: even if the agents are secured from one another, a host may allow some other agent to retrieve the agent's information. Other agent may tamper with the given agent on the permission of the host. Bottom line is if you want your agent to fully secure you should always send it to the trusted host.

3.4 Security Model

This section describes the security model that provides an overall framework for aglet security as is shown by the Luca Ferrari in its aglets manual [8].

3.4.1 Principals

Principals in agent system are authenticated identities that are used to enforce the policies that are defined by authorities and to authenticate the developer of the program or the host it is communicating with.

- Aglet: as they are autonomous in nature, it is reasonable to assume that they can define their own security policies. There are three roles for aglet principles:
  - Aglet: an aglet object is the thread responsible for executing the aglet.

- Aglet manufacturer: the aglet manufacturer represents the person or organization that implemented the aglet program. The behavior with proper permissions is also set by these manufacturers. Aglet owner: the aglet owner represents the person or organization that launched the aglet. Because the owner is responsible for its aglet, this principal is used for authorization of the aglet.

- Context and Server: contexts and servers are responsible for keeping the underlying operating system safe by protecting it from malicious aglets. A server defines a minimal security policy to protect local resources. On the other hand, each context is responsible for hosting visiting aglets, and it enables the access to various local resources. There are three roles for context and server principals:
  - Context: a context is a place that hosts aglets.
  - Context manufacturer: this is the manufacturer of a context server. As with aglets, it is in a manufacturer's concern that no one be able to claim damage caused by a malfunctioning context and server.
  - Context owner: the context represents the context owner. This principal is used for authenticating hosts in the role of sender and receiver of aglets.
- Network Domain: the domain can be represented as the group of servers. The principal of the domain authority is used to authenticate whether a server is a member of its domain or not. A network domain is responsible for keeping its network secure so that all incoming aglets can complete their tasks safely.

3.4.2 Permissions

Permissions define the capabilities of executing aglets by implementing the access restrictions and limits on resource usage. Permission is a resource, such as a local file, together with appropriate actions such as reading or writing a file, listening to a network port, or creating a desktop window. Several permissions are available for aglets [8]:

- File permission: Access to the local file system is also subject to control. The aglet can be granted access to a specific file or an entire directory.

  FilePermission "/tmp/*" "read"

FilePermission “c:\public*” “read”

- Network permissions: the aglet can be granted access to a specific host or to listen on a specific port.
  SocketPermissions “atp://172.17.9.54:4434” “connect”

- WindowPermission: an aglet can be granted permission to open a window.
- ContextPermission: an aglet can be given permission to use services provided by the context. This may include access to methods for creating, cloning, dispatching, retracting, deactivating and activating aglets.
- AgletPermission: the methods provided by individual aglets also need some control. An aglet can be allowed to invoke methods in another aglet owned by a principal given by a name.

3.4.3 Protections

Although an aglet may be granted access to a resource or other aglets, it may also want to protect itself from access by other entities. For example, it is reasonable for you to request that an aglet should be disposed by you only, whereas other methods may be publicly accessible.

3.4.4 Policy and Authority

All the terms that we have discussed in section 3.4.1 to section 3.4.3 may define security policies. A security policy is created as a set of rules denoted by principals and privileges. It is a policy authority that defines these policies. A policy authority is the person or organization for resources consumed by other entities. In this security model there are three authorities:

- Aglet owner: as the aglets are autonomous in behavior, it is reasonable to assume that they can define their own security policy. The main objective of the aglet owner is to protect the aglet from attacks. The aglet owner defines the security

policies for aglet. When an aglet migrates to remote host or a context, it requests the host to implement the security policy.

- Context owner: a context authority keeps the server and its system safe from malicious agents. It defines the action that an aglet can take in a particular context.
- Network domain owner: network domain is responsible for keeping its network of system secure so that aglets can securely execute and finish their task

3.5 Agent Characteristics of Java: Benefits

Java is a programming language which is used to write the softwares. It is open source and freely available. It has some benefits which makes it one of the best language for writing the agents. Java is an object-oriented and network oriented language. It also provides support for the mobility though it's weak mobility by using the concept of serialization. This serialization mechanism helps the instance variables to maintain their values.

The tutorial on Java available with the SDK (Software Development Kit) shows some properties of Java programming language which makes it a good language for mobile agent technology [9]. These are:

- Platform-independence: java is designed to work in heterogeneous environment, it compiles the application and converts it into Java byte-code which is architecture neutral code, that can run on any system supporting JVM (Java Virtual Machine). This facility of running the Java byte-code anywhere on the system makes it platform-independent language. Secure execution: java is secure language which makes it well suited for the use on Internet and intranets. Java doesn't support the pointers so it eliminates the possibility of overwriting the memory and corrupting data, it doesn't allow illegal type casting too, proper access permission are given as private, public and protected data member which prevents most activities of virus attack. And if somebody tried to alter the byte code information, java byte

code verifier enable that it doesn't violet the basic semantics of Java. These types of facilities provided with Java makes it a secure language to work on Internet or over networks.

- Dynamic class loading: this dynamic class loading mechanism enables the virtual machine to load and define the classes at runtime. It provides the separate and protective name space where agents execute independently and safely from each other.
- Multithread programming: agents are autonomous in nature i.e. they can execute independently of other agents. It implies that each agent executes in its own lightweight process or thread. This kind of multithreading is very well supported by the Java where each agent executes in its own thread. Java also supports the synchronous primitives that enable agent interaction.
- Object serialization: serialization and deserialization is an important feature in mobile-agent. Java has a built-in serialization mechanism which serializes the details of the objects that can be reconstructed later after migration. This serialization is supported by Java to help the agents make mobile.

3.6 Agent characteristics of Java: Drawbacks

As we have seen the benefits of Java language in the creation of mobile-agents and we have seen it is highly suitable language for the creation of mobile agents. There are still some drawbacks of using this language too, which should not be overlooked. Some of the shortcomings of Java can be worked out but other are more serious, these drawbacks are:

- Inadequate support for resource control: Java language doesn't provide any means to control over the resources consumed by the Java objects. It means that if an agent starts looping indefinitely, it will waste processor cycles and start consuming memory resources. This will give rise to the denial-of-service attack. If an agent keeps consuming the resources it will make impossible for the owner to control over its resources. Unfortunately, Java provides no way to control the limit of processor and memory usage allocated to an object.

- No protection of references: A Java object's public methods are available to any other object that has a reference to it. There is no way that an agent can monitor or control over the agents that are accessing its methods.
- No object ownership of references: the collection of the references of the Java objects is a task of the garbage collector. So there is no owner of the objects references. It implies that even if you have killed the agent but still its reference is alive and there is no explicit way that you can destroy it, so as to save it from the third party.
- No support for preservation and resumption of the execution state: Java only supports the weak mobility concept, as it is impossible in Java to retrieve the execution state of the program, and migrate it to the next host. This execution state information is stored in the stacks area where Java has no reach. So to provide the strong mobility concept, in which an agent can resume its execution where it stopped on another host, can not be achieved by using Java language and we have to depend on other language to fulfill this requirement.

3.7 Available Mobile Agent System

There are number of mobile-agent system available in Java. Some are under development and some are available on Internet.

We have shown few mobile agent systems:

- JADE (Java Agent DEvelopment framework) [10]: JADE is an agent development framework which simplifies the development of agents. It provides the basic infrastructure and the services for its development. Jade is implemented in Java so it can run on all JVM. It is also FIPA compliant. Jade provides an agent platform, directory facilitator and agent communication channel which eases the agent's development and its functionality over the network.
- Odyssey: General Magic Inc. invented the mobile agent and created the first commercial mobile agent system, called Telescript [11]. As it was based on

proprietary language, it had a short life. It provides a Java class library which facilitates the developer to create their own mobile agent application.

- Concordia: Mitsubishi's Concordia is a framework for the development and management of mobile agent applications for any application that supports Java [12]. A Concordia system consists of a server and a set of agents. Like aglets, Concordia also provides the security manager which enables the secure execution of the agents.
- Voyager: ObjectSpace's Voyager is a platform for agent-enhanced ORB (Object Request Broker) for Java [13]. Voyager provides a set of object messaging capabilities and it also allows the objects to move in the network as agents. Its drawback is that it provides no security management for un-trusted agents.

All the java-based agent systems have so many things in common. They all use java virtual machine and java serialization concept. It only differs in their transport mechanism and their way of messaging.

3.8 Summary

In this chapter we have discussed about Aglets mobile agent system. After that we have taken an overview of the aglet security model. This model supports the definition of security policies and describes how and where a secure aglet system enforces these policies. We then discuss briefly discussed about the Java programming language, about its popularity in creation of many mobile-agent systems, its drawbacks while mentioning agents and finally we discussed about the other java-based mobile agent system apart from Aglets. In the next chapter we will look at the security threats on the mobile agents and the ways to countermeasure these attacks, we will focus on the malicious host attack that can be performed on the arriving agent.

Chapter 4
Security Threats & Countermeasures

Mobile agents can travel to other system in a network and can execute there by consuming remote host resources, this is the basic property of Mobile agents and because of this property the Mobile agents are open to several attacks and abuses. Mobile agents amplify the threats of abuse and misuse due to their mobility and execution on different platform. This chapter discusses about the threats that can be encountered by mobile agents. Firstly we will discuss the classification of all the threats and then we discuss these threats in detail and finally the countermeasures taken to prevent or avoid these attacks.

4.1 Security Threats

Internet has become the main revolutionary medium for expanding business and as Internet is expanding, more and more companies in corporate world want to take full advantage of it. People now want to buy and sell items instantly, they don't want to go to the shops and look for their item but instead they are using Internet for this, where they can find the desired item just on a mouse click. People are now interested in electronic business, where they can do necessary transactions on Internet only. Now as their

interaction with the Internet increasing, they need new technology which can support their secure transaction. Here mobile agent technology gives answers to the problems faced by the people. This technology gives the flexibility to the users to expand their transactions online without their direct intervention. As the mobile agents are mobile and they can roam in the network away from the owner's site. It increases its risk to get hacked and misbehaved. These agents can attack each other and are themselves vulnerable to many attacks. Security issues are the main obstacle to implement the mobile agents. Until these security issues are addressed carefully, this useful technology can not be used to its full potential.

Without security and cryptography e-commerce can not work in the real world. An agent is safe in its owner site but when it ventures out and gets executed on some other platform, it becomes vulnerable to many attacks. A host may be malicious in various ways, it may perform passive attacks, where it can try to intercept the agent's communication and steal the messages, or more active attack where a host tries to delete the agent, modify some part of the agent's code, feeding some false information or send it to some false host or not send at all.

The malicious host that is doing alteration to the agent's code and data, if not detected immediately may not be tracked down later. So there should be some method or technique which can stop or delay this alteration.

According to Mousa Alfalayleh and Lijiljana Brankovic, the threats in mobile code systems can be broadly classified as [14]:

1. Threats that originate from an agent which attacks on another agent platform. This kind of attack can be protected in many ways e.g. sandboxing, software fault isolation, proof-carrying code, operating system access controls.
2. An agent attacking another agent on the agent platform, protection against this attack has been made as agent separation implemented on hosts.
3. An agent platform attacking an agent, protection of agents from malicious hosts remains a major problem in agent technology.

Major problem is how to secure an agent's private information which it is carrying along with it, the messages it is generating on the remote host and finally how to secure itself from unwanted alteration.

Here in this project we are not going to discuss the first two classes of attacks, but we will concentrate more on final part where we have to deal the problems when agent is out of the reach of its owner and executing on some remote host.

4.2 Malicious Host Problem

Malicious agent problem have several solutions and the problem may seem easy to solve, but protecting an agent from malicious host is a difficult problem or at first sight seems impossible to solve. If an agent is executing on a host other than the owner, it is totally under the mercy of the remote host as it can do anything with it.

Here we have to find out, how an agent can do secure electronic transaction while executing on remote host.

W. Jansen and T. Karygiannis have discussed some threats which can occur due to malicious host [15]:

4.2.1 Eavesdropping: eavesdropping threat involves the monitoring and interception of the secret information which is being communicated between authenticated hosts. In the case of malicious host problem, where an agent is executing on the remote host and the host has every chance to monitor each instruction executed by the agent. This becomes more serious problem in case of malicious host. If the host gets access to agent code, it can expose proprietary algorithm, private information, negotiation strategies or some secret information of the agent.

4.2.2 Alteration: alteration attack is extended attack of eavesdropping threat that we have mentioned above. If the agent is executing on malicious host it is exposing its code and data. A malicious host can change the data or the behavior of the agent. This type of

attack can be detected by having the agent signed by original author. The change in the agent's state during its execution or the change in the date while executing on the malicious host still does not a general solution.

If an agent is moving to several hosts on its itinerary and one or more hosts turned out to be malicious, then this kind of attack is nearly impossible to track down in the end as the agent has undergone numerous changes of its data or code.

The security risks arises while moving agent from its home platform to another is referred to as the "single-hop" problem, while the security risks resulting from an agent visiting several platform is referred to as the "multi-hop" problem. The risks in the case of single-hop problem are easy to alleviate than to the multi-hop scenario.

4.2.3 Masquerade: an agent platform can masquerade itself as another platform, to make itself appear to be an authenticated one, to a mobile agent. It may attract the agent to come and make it execute so that malicious host can extract sensitive information. These kinds of malicious platforms can harm both the agents and the platform whose identity they have assumed. These types of masquerading platform can give rise to the other attacks like eavesdropping and alteration.

4.2.4 Denial of Service: this kind of attack occurs when an agent comes to a host to produce some results by utilizing the resources of the host. Now a malicious host can ignore the agent request, may introduce delays or it may not allow the agent to execute at all. This kind of attack may give rise to deadlock condition or unnecessary long delays when multiple agents are dependent on each other's result.

4.3 Counter Measure

In this section we will discuss the general solutions to the malicious host problem. As the mobile agents are traveling with code and data, we will first try to avoid the problem, if not, then we will see how to handle them from those proposed solutions.

4.3.1 Avoiding the problem: this is the problem in which an agent is avoided to get executed on the untrusted remote host. An agent is sent to only those hosts which are authenticated one. This resembles the same problem in real life where interaction with the organization or humans with the bad reputations is avoided. So all the trusted hosts form a closed group of network where an agent can execute and agent doesn't go anywhere else to get executed apart from the trusted network. This somehow restricted the usefulness of mobile agents, as agents are more useful in case of open agent system scenario where any number of agent system can attach themselves to the network to provide its services. The problem with this kind of approach is that it is not clear in advance which host to trust or not. This is very conservative approach and severely reduces the number of hosts on which an agent may migrate to [16].

4.3.2 Hardware-based Security: Hardware-based security is the most effective approach but it is not feasible enough. This approach requires installing a secure hardware on each node where a mobile agent can migrate to and execute. This is again meant for the small network where such procedures are possible.

4.3.3 Encryption-based: another approach is to use mobile cryptography techniques. Symmetric cryptography is well suited for those mobile agents which need to send the results back to its owner. Here in this case various cryptography techniques can be used to encrypt the results it created and also to encrypt the keys which are used to encrypt the data that produced on the remote host. It gives security to eavesdropping and alteration attacks performed to the agents on remote host. Here similar techniques authorization and authentication are also mentioned by M.Farmer, J. Guttman, and V. Swarup [17].

4.3.4 Computed with encrypted functions: sander and tschudin have proposed a method whereby an agent platform can execute a program embodying an enciphered function without being able to discern the original function [18]. It says that a function can be encrypted in such a way that they can still be implemented as programs. The resulting program can be understandable by processor but processor will still wont understand the program's function.

The problem is stated by Sander and Tschudin a follows:
"Alice has an algorithm to compute a function *f*. bob has an input *x* and is willing to compute *f(x)* for her, but Alice wants bob to learn nothing substantial about *f*. moreover, bob should not need to interact with alice during the computation of *f(x)*."
This idea, if implemented successfully, can solve most of the problem in mobile-agent. But he challenge is to find the appropriate encryption scheme that can do transformation to the functions. There is still a lot of work to be done to explore such possibilities.

4.3.5 Co-operating agents: by using cooperating agents the information and functionality can be divided among various agents. It gives benefit that if somebody tries to compromise one agent, may not be able to extract the complete information. In the case of selling-buying agents, several mobile agents can be used and they can choose for the best offer after their votes amongst each other.

4.3.6 Execution tracing: execution tracing is used to detect unauthorized alteration of an agent by recording its execution detail at every platform. In this case each platform is supposed to maintain a log file of the execution detail of the agent on that system. Now the drawback is to maintain the various log files produced.

4.3.7 Obfuscated code: Hohl has proposed code mess-up technique in which an agent's code is scramble in such a way that no one is able to gain a complete understanding of its function [19]. There are few other authors who published there own techniques of obfuscation [20]. However, there is no general algorithm or approach exists for providing blackbox security. One serious drawback of this scheme is the difficulty of quantifying the protection time provided by the obfuscation algorithm.

4.4 Summary

The security issues for non-mobile agents can be tackled to a great extent through existing cryptography and protocols. But when we talk about the mobile-agent we have to

tackle them differently. Here in this chapter we have seen the problem related to mobile agent and the proposed solutions for the problems. Although there is no general solution for the main problem of mobile agent protection, we have solutions which can provide security to the agent at least partially against a malicious host. In the next chapter we will see the how the security issues are implemented in a mobile agent application by using various techniques of cryptography and obfuscation.

Chapter 5
Implementation Work

This chapter deals with the implementation work which is done to show the various aspects of security in mobile agents. In this chapter we will discuss about the various ways through which the security of the agents, which are executing on remote host, are performed. In this chapter we will take an example of seller-buyer agent and we will discuss the attacks which it can encounter while traveling to remote host to get its work done.

5.1 Implementation of Mobile Agents

In this chapter we will take an example of mobile agent application, where an agent tries to buy some goods from the remote host and we will see how many types of attacks it becomes vulnerable to. Here we are considering some facts like:

- We are imitating an application of e-commerce where mobile agents help in selling and buying the desired item from the behalf of their owner.
- Agent is roaming in a trustful environment, where there are less chances of a host to get compromised.

- Here we have to set an itinerary of an agent, it consists of all the hosts where the agent is supposed to move, one by one or you can say the path of the agent where it has to proceed to get its work done.
- When this agent finds a host where it can get its item at the desired cost, it will come back to the owner with the information about the host and the prices offered by the seller host.
- Here we are considering the network to be trustworthy but still our agent has to be cautious and prepared that what if a host turns out to be compromised, then how it will protect its private data, protects the results it is going to produce on remote host and finally how it will protect itself.

5.2 Application

We have shown it with the help of an example-application consists of Buyer-agent GUI (Graphical User Interface) which inputs some data from user and uses that information for doing its task.

The tests are done on following specification:

1. Windows XP (Professional) Platform
2. Aglet Software Development Kit (ASDK) 2.0.2.
3. Java Development Kit (JDK) 1.5.
4. ProGuard Application Programming Interface (API) 4.

Here fig. 5.1 shows the GUI of Buyer agent. It has some fields as:

- Destination: it shows the itinerary of destination where it has to migrate on the network. The list appears on clicking on the button AddressBook.
- Address: this field shows the next host address where the agent will migrate to.
- Amount: this has the e-money of the agent which it is carrying with it. This is the private information which agent is carrying while traveling.
- Items to buy: this field specifies the item to be bought by the agent.
- No. Of items: this field tells the number of items to be purchased.

- GO button: clicking this button starts the migration of the agent to the host address specified in the Address field.
- Send Clone button: this button sends the clone of the agent to the machine.
- Dispose button: this button disposes the agent.
- Text Area: the text area on the agent GUI shows the results taken by the agent back to the owner.

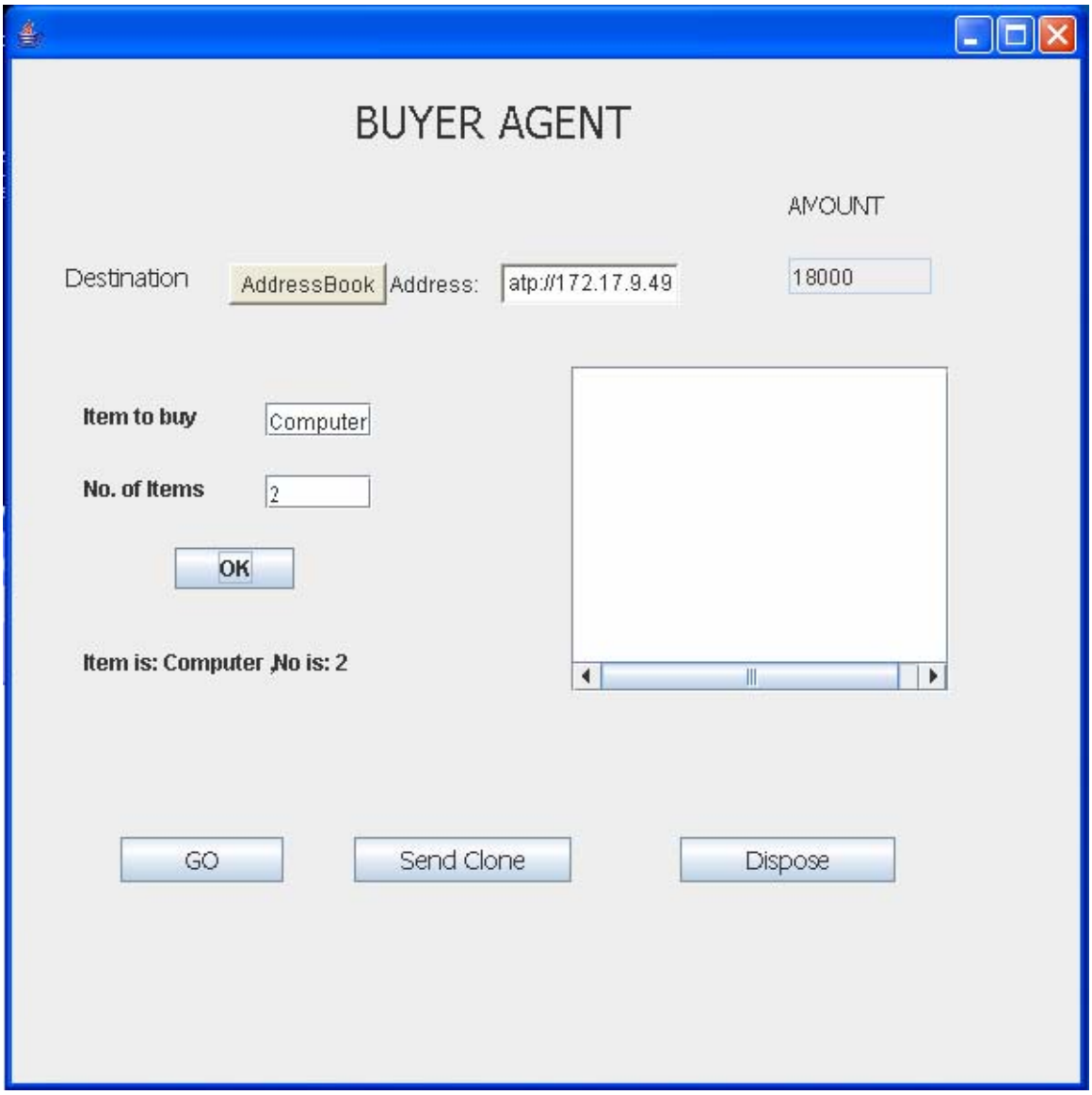

Fig. 5.1

This fig. 5.1 shows the GUI of the buyer agent.

Seller agent is an agent which is present on the remote host and whose information is supposed to take back by the buyer agent to its owner. Information like where it is present and the cost of the item set by the seller-agent host.

To be able to recognize by the buyer-agent, each seller-agent has to set its properties.

Fig. 5.2 shows the command prompt of the interested remote host, who want to sell some goods. These remote hosts have to set the property of their agents through the keyword like "SellerBuyer", which reflects the purpose or the type of item it is dealing in the network market.

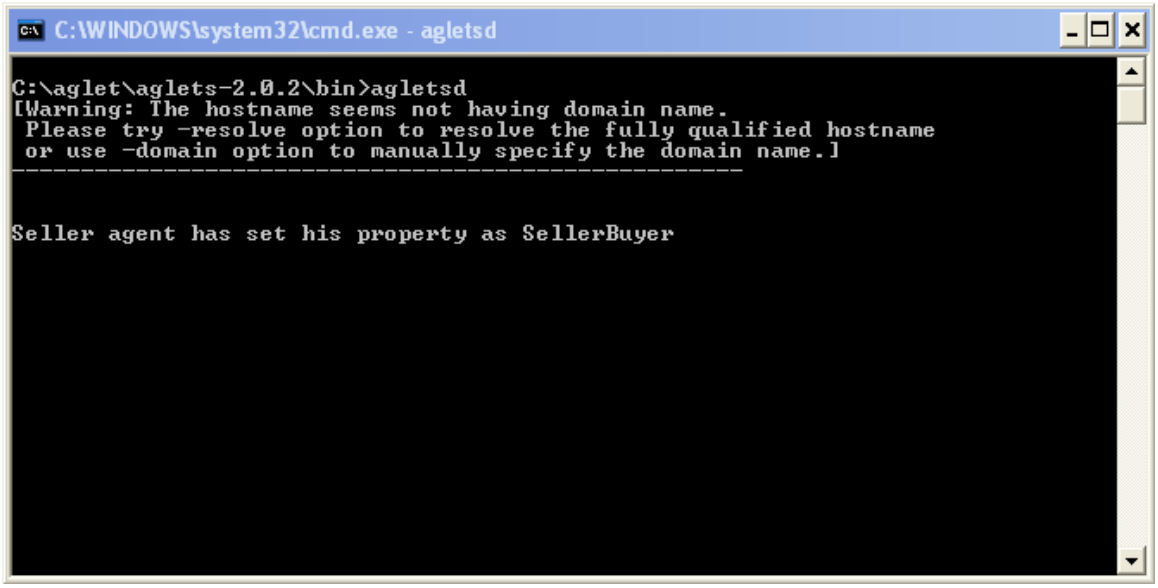

Fig. 5.2

5.3 Security Issues

This section shows the security issues that can be faced by this application. The security problem can be:

- Protection of the private data of the buyer-agent.
- Protection of the results generated by the agent.

- Protection of the agent itself on remote host.

5.4 protection of agent

Protection of this agent application can be provided by following countermeasures.

5.4.1 Cryptography

Protection of the private data of the buyer-agent: The private data may include credit card information or e-cash, which can be changed or used inappropriately by the remote host which is executing a mobile-agent. We are using the conventional technique of symmetric-key cryptography where the private data of the buyer agent, which is the e-money that it is carrying, is sent in the encrypted form. This encrypted private data will remain intact until its secret keys are compromised.

Protection of the results generated by the agent: When a mobile-agent executes on some host, it may produce some data that may be confidential to the owner. That result can be used or altered by the same host where it is producing the results. This kind of attack is prevented by using symmetric as well as asymmetric keys. Where the result created by buyer agent is first encrypted using the symmetric key or secret key which is produced on the remote host and then this secret key is encrypted afterwards using asymmetric key [21]. Then these two encrypted data is sent back to the owner, the encrypted key and the encrypted results. This data is decrypted back at the owner's machine. This way the security of the results can be achieved.

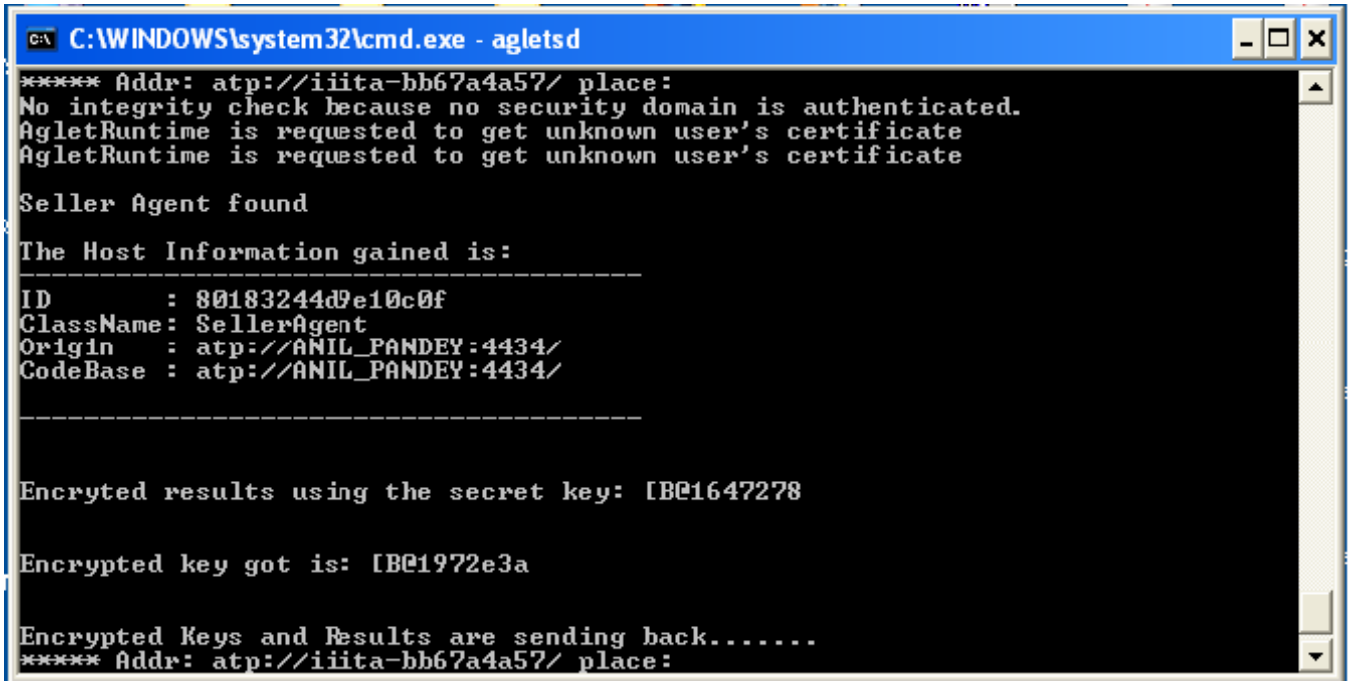

Fig. 5.3

Here figure 5.3 shows the results found on the host. It is showing few points as:

1. Seller agent is found. This means the seller agent is found on the host which is showing interest in "SellerBuyer" property.
2. Buyer-agent gains the information of the host and the cost of the items it is selling.
3. secret key is produced and the result are encrypted from this key.
4. Finally the secret key is itself encrypted using the owner's public key.
5. Last line in the fig 5.3 shows that the results and keys are sent back to the owner.

Figure 5.4 shows the results on the owner's side.

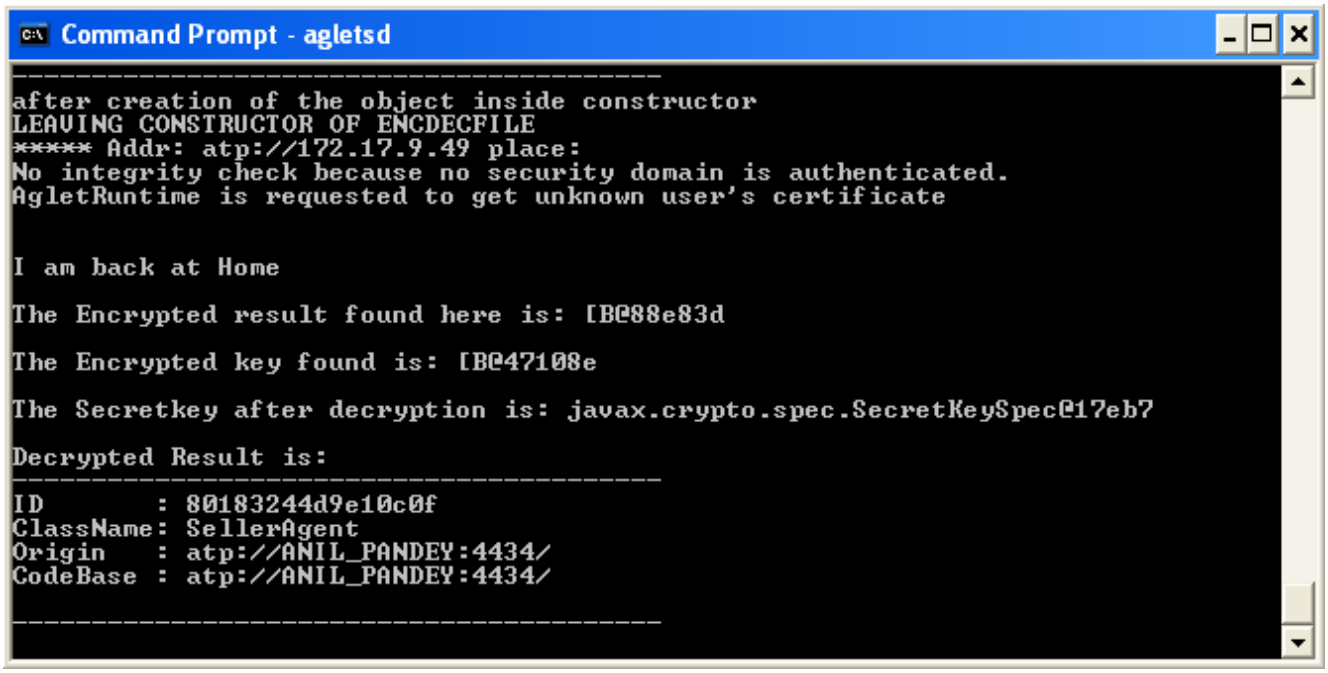

Fig. 5.4

This fig 5.4 shows the results found on the owner side. It shows:

1. The encrypted results that are found.
2. The encrypted secret key that is decrypted using owner's private key.
3. That secret key is further used to decrypt the results.

5.4.2 Obfuscation

Protection of the agent itself on remote host: Obfuscation is an approach in which code is transformed or messed up in such a way, it becomes hard to understand, but performs the same function as the original program. Collberg, Thomporson and Low [21] have shown some classification of obfuscation technique.

Here according to Collberg, Thomporson and Low the obfuscation can fall into few categories, these are:

1. Layout Transformation: it is a transformation with low potency as there is very little semantic formatting is done. So no great confusion is created. Scrambling of identifiers name is a way to do free transformation of this type.
2. Control transformation: control transformation randomizes the order in which computation are supposed to carry out. It may insert new code or dead code or make algorithmic changes to the source application.

Now here are some examples to show the results came out from obfuscation. Here we applied obfuscation over java class files. These class files are decompiled by Cavaj Java Decompiler. The results are like the following:

Original program:

```
import java.applet.Applet;
import java.awt.*;
import java.util.*;

public class Blink extends Applet
{

    private Timer timer;
    private String labelString;
    private int delay;

    public Blink()
    {
    }

    public void init()
    {
        String s = getParameter("speed");
        delay = s != null ? 1000 / Integer.parseInt(s) : 400;
        labelString = getParameter("lbl");
        if(labelString == null)
        {
            labelString = "Blink";
        }
        Font font = new Font("Serif", 0, 24);
        setFont(font);
    }

    public void start()
    {
        timer = new Timer();
        timer.schedule(new TimerTask() {
```

```
            final Blink this$0;

            public void run()
            {
                repaint();
            }


            {
                this$0 = Blink.this;
                super();
            }
        }, delay, delay);
    }
```

Obfuscated program:

```
import java.applet.Applet;
import java.awt.*;
import java.util.StringTokenizer;
import java.util.Timer;

public class Blink extends Applet
{

    private Timer HG_timer;
    private String GG_labelString;
    private int FG_delay;

    public Blink()
    {
    }

    public void init()
    {
        String s = getParameter("speed");
        FG_delay = s != null ? 1000 / Integer.parseInt(s) : 400;
        GG_labelString = getParameter("lbl");
        if(GG_labelString == null)
        {
            GG_labelString = "Blink";
        }
        Font font = new Font("Serif", 0, 24);
        setFont(font);
    }

    public void start()
    {
        HG_timer = new Timer();
        HG_timer.schedule(new _cls1(), FG_delay, FG_delay);
    }
private class _cls1 extends TimerTask
```

```
    {

        final Blink IG_this$0;

        public void run()
        {
            IG_this$0.repaint();
        }

        _cls1()
        {
            IG_this$0 = Blink.this;
            super();
        }
    }
```

As we can see here the obfuscated program has changed the variable names from the confusing one. The confusion of the names can be increased depending on the user's choice.

This sort of obfuscation is useful when the program size lies between large to very large.

The complex obfuscation here can make the program look worse and very difficult to understand. This can be shown using a small example:

Original program:

```
public class HelloWorld
{
   public static void main(String argv[])
   {
   System.out.println("Hello World!");
   }
}
```

Obfuscated program:

```
import java.io.PrintStream;

public class HelloWorld
{

    private static String _0;
```

```
    public HelloWorld()
    {
    }

    public static void main(String args[])
    {
        System.out.println(_0.substring(0, 12));
    }

    static
    {
        "cNGGD\013|DYGO\n".toCharArray();
        0;
_L2:
        JVM INSTR dup2 ;
        JVM INSTR dup2 ;
        JVM INSTR caload ;
        0x2b;
        JVM INSTR ixor ;
        (char);
        JVM INSTR castore ;
        1;
        JVM INSTR iadd ;
        JVM INSTR dup ;
        12;
        JVM INSTR icmplt 6;
           goto _L1 _L2
_L1:
        JVM INSTR pop ;
        String.copyValueOf();
        _0;
        0x1139b2fb1b2L != System.currentTimeMillis();
        0x1134df04db2L != System.currentTimeMillis();
_L4:
        JVM INSTR dup ;
        JVM INSTR ifgt 43;
           goto _L3 _L4
_L3:
        JVM INSTR pop ;
_L6:
        JVM INSTR dup ;
        JVM INSTR iflt 48;
           goto _L5 _L6
_L5:
        JVM INSTR pop ;
        return;
        throw ;
    }
}
```

The output produced by both the programs is: “Hello World!”.

Here the level of obfuscation is high, this obfuscation made the program look difficult to understand. This kind of obfuscation is useful to hide the important algorithms.

Here in our example, we obfuscate the class files related to our Buyer Agent before sending it to the remote host. So by doing this we are making our agent more secure as it is using both the techniques of cryptography and obfuscation for its protection. This gives us few thoughtful points as:

1. If the attacker is able to get the code of the agent, he will look for the private data which is been encrypted. So this data is protected as far as he compromised the secret keys.
2. On obfuscating the whole agent code, it will make it more difficult for the attacker to understand the code also obfuscation makes private data look more ordinary. So it will take attacker much more time to crack the agent and its private information.

5.5 Summary

In this chapter we displayed the snapshots of our cryptography and obfuscation results. This chapter shows the combination of cryptography and obfuscation may produce better results and obfuscation can confuse the attacker for some time depending on the complexity of the algorithm used for obfuscation.

Chapter 6
Conclusion and Future Work

6.1 Conclusion & Future Work

In this thesis we have discussed about the mobile-agents, its security issues and its countermeasures. Mobile agent system is a very promising paradigm, it has shown its presence in many applications like distributed networking, e-commerce applications etc. There are numerous advantages of using the mobile agent paradigm rather than conventional paradigm such as client-server based technologies. In one way it provides the abstraction to networking. The benefits of mobile-agent technology can not be exploited fully until all the security issues are properly addressed. We try to provide a better solution by combining the results of two or more solutions. This solution addresses most of the problem but still it is very much dependent on the complexity of the algorithm used and the possibility that how soon the professional hacker can de-obfuscate the program. It still does not address the problem of denial of service. Also obfuscation technique affects the performance of the code as it may change the whole layout of the program structure. So to keep our application safe, new algorithms have to be developed with time. But providing the complete solution to mobile agents is still a far way to go. Lot of research is required in this area.

Future work may include following points:

1. We have used these techniques over a network where the agent is roaming freely and the security is applied over that application. These security issues can be checked by applying for the e-commerce application on different networks and over the Internet.
2. Further new API for obfuscation can be used to make the obfuscation more difficult to de-obfuscate. This technique of combing two or more solution can be applied to different combinations of the proposed solutions.

REFERENCES

[1]	Object Management Group, NEEDAM MA, Agent Technology Green Paper 1 Sept 2000, PP. 8-11.
[2]	Hyacinth S. Nwana, "*Software Agents: An Overview*", Intelligent systems Research, Advanced Applications & Technology Department, Ipswich, Suffolk U.K. Cambridge University Press 1996.
[3]	The Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org/.
[4]	James Odell, "*Objects and Agents: How Do They Differ*?".
[5]	Danny B.Lange and Mitsuru Oshima, "*Programming and Developing Java Mobile Agents with Aglets*". (Addison Wesley publication).
[6]	Colin G. Harrison, David M. Chess, and Aaron Kershenbaum, "*Mobile Agents: Are they a good idea*?", technical report, 1995, IBM Research Division.
[7]	www.trl.ibm.com. Aglets.
[8]	Luca Ferrari, "*The Aglets 2.0.2 User's Manual*", October 2004.
[9]	Java Programming Concepts From The Tutorial, http://java.sun.com/docs/books/tutorial/java/index.html.
[10]	JADE, http://jade.tilab.com.
[11]	General Magic Inc. Odyssey http://www.genmagic.com/agents.
[12]	Reuven Koblick: Mitsubishi's Concordia, March 1999

[13]	ObjectSpace's Voyager http://www.recursionsw.com/voyager.htm
[14]	Mousa Alfalayleh and Lijiljana Brankovic, "*An Overview of Security Issues and Techniques in Mobile Agents*", The School of Electrical Engineering and Computer Science, The University of Newcastle, Newcastle, NSW 2308, Australia.
[15]	W. Jansen and T. Karygiannis, "*Mobile Agent Security* ", Nist Special Publication 800-19 -, 2000. National Institute of Standards Technology.
[16]	Joris Claessens, Bart Preneel, Joos Vandewalle , "*(How) Can Mobile Agents Do Secure Electronic Transactions On Untrusted Hosts? A Survey Of The Security Issues and The Current Solutions*", pp. 38-41, ACM Transactions on Internet Technology, Vol. 3, No. 1, February 2003.
[17]	William M.Farmer, Joshua D. Guttman, and Vipin Swarup, "*Security for Mobile Agents: Authentication and State Appraisal*", pp. 5-11, European Symposium on Research in Computer Security (ESORICS).
[18]	T. Sander and C. F. Tschudin, "*Protecting Mobile Agents Against Malicious Hosts*". G. Vigna, editor, Mobile Agents and Security, volume 1419 of LNCS, pp. 44–60. Springer-Verlag, June 1998.
[19]	F. Hohl, "*Time Limited Blackbox Security*". G. Vigna, editor, Mobile Agents and Security, Volume 1419 of LNCS, pp. 97-102. Springer-Verlag, June 1998.
[20]	T.W.How, H.Y.Chen and M.H. Tsai, "*Three Control Flow Obfuscation Methods For Java Software*", The Institution of Engineering and Technology 2006, IEE Proceedings online no. 20050010.
[21]	R.L.Rivest, A.Shamir, And L.Adleman, *"A Method For Obtaining Digital Signatures And Public-Key Cryptosystems*", February 1978 Communications of the ACM, Volume 21 Issue 2.

[22]	C. Collgerg, C. Thomborson and D. Low, "*A Taxonomy of Obfuscating Transformations*", Department of computer Science, The university of Auckland, New Zealand, Technical Report #148.

APPENDIX A

AGLETS SOFTWARE DEVELOPMENT KIT

The mobile agent system used in this work is aglet software development kit (ASDK) 2.0.2. It is free software. Aglets software development kit (ASDK) was originally developed at IBM Tokyo Research Laboratory but currently the project is hosted at Sourceforge and can be downloaded from http://sourceforge.net/projects/aglets/.
The installation process of Aglets is given in user manual which can be downloaded for free from the above link.
When the installation process is done, we can run the Aglets server called Tahiti which prompts for login name and password as shown in the Fig. A.1. default login and password is shown in ASDK user's manual.

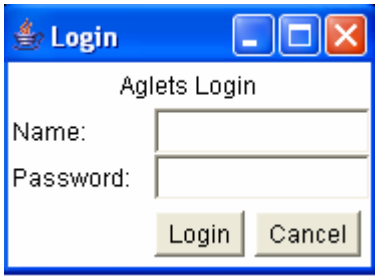

Fig A.1

After Login, Tahiti Server started which supports the aglets to run inside it and wherever this server is installed it can support execution of the aglet.
The Tahiti server is used to Create, Dispatch, Clone, Retract or Dispose the agent. Figure A.2 shows the Tahiti Server.

Fig A.2

Aglets are java class files which can be run from this server by clicking the Create Button on the main menu of GUI.

The Figure A.3 shows the Create Aglet window which is used to start an aglet.

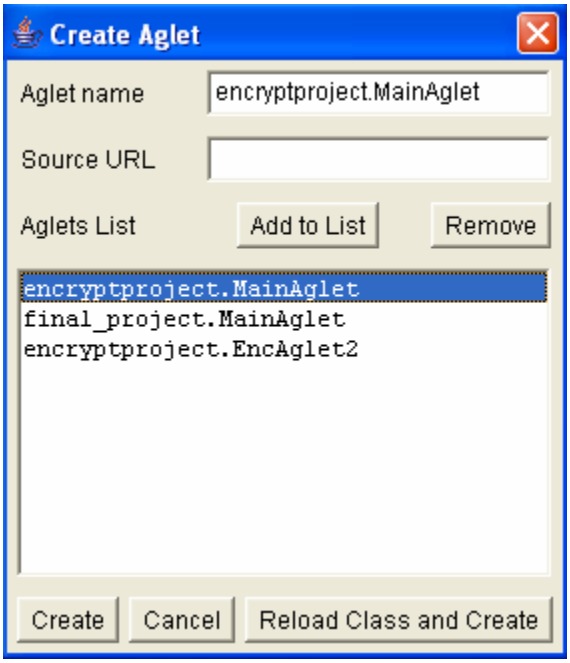

Fig A.3

On creating the agent, the Aglet Viewer shows the aglets currently in the running mode. Fig A.4 shows the aglet running on the local machine.

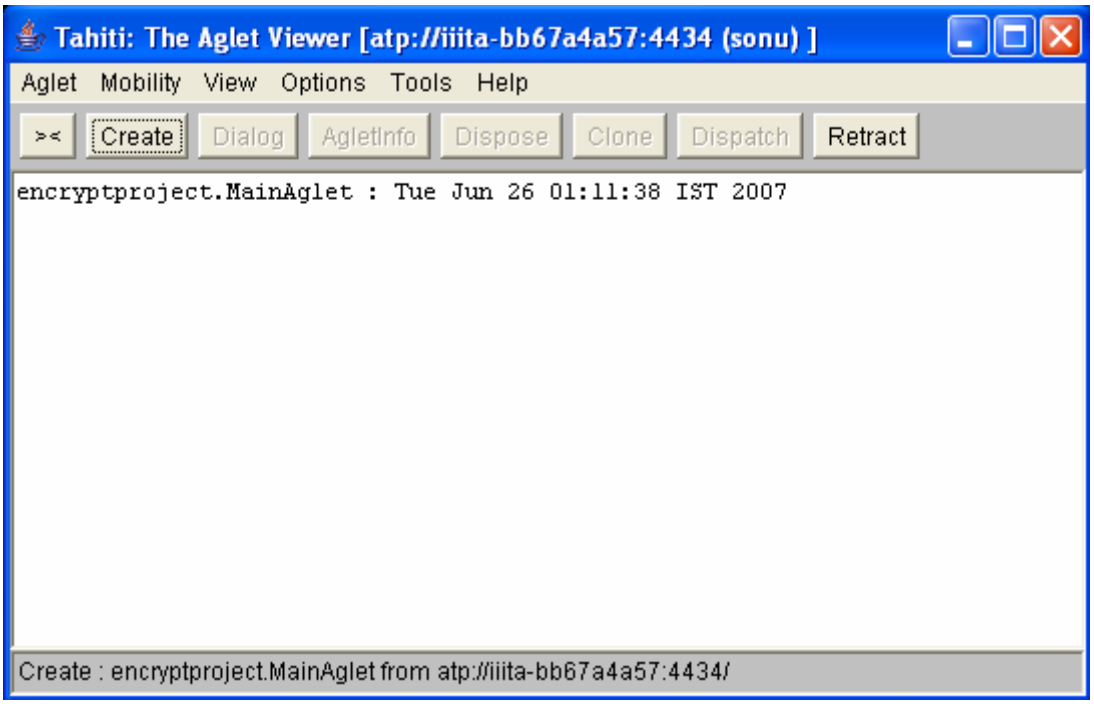

Fig A.4

The status field at the bottom shows the recent event that occurred, here an aglet named 'MainAglet' was created and hence the message 'Create : MainAglet'.

To dispatch this aglet the remote site must be running the Aglet Server Tahiti. By giving the proper security permissions the aglet can be dispatched to the remote host.

APPENDIX B

ProGuard 4.0 (Obfuscation API)

Progurad Version 4.0 is a free Java class file shrinker, optimizer, obfuscator and preverifies. If used as obfuscator it renames classes, fields, and methods using short meaningless names.

Proguard API (Application Programming Interface) can be included to the java application and can be used to obfuscate the programs. This API can also be used for Ant and for the J2ME Wireless Toolkit.

Performance **Table B.1** on predefined libraries:

Input Program	**Original Size**	**After shrinking**	**After optimization**	**After obfuscation**	**Total reduction**	**Time**	**Memory usage**
ClassPath, the GNU runtime library	2.0 M	2.0 M	1.9M	1.9M	4%	31s	39M
ProGuard itself	404K	364K	361K	231K	42%	21s	35M

Table B.1

* M=MegaByte, K=KiloByte

Results were measured with ProGuard 3.0 on a 2.6 GHz Pentium 4 with 512 MB of memory, using Sun JDK 1.4.

Using ProGuard:

- The manual is provided at the time of downloading the proguard. It tell all the steps to download this API.
- Before using ProGuard the classpath should be set in system environment variables for the given API.

Proguard can be used on command prompt. Configuration file has to set where the path of the Input or Output jar/war/class files is mentioned. On running the ProGuard jar, this configuration file will be read to fetch for the input and output.

Figure B.1 shows the results:

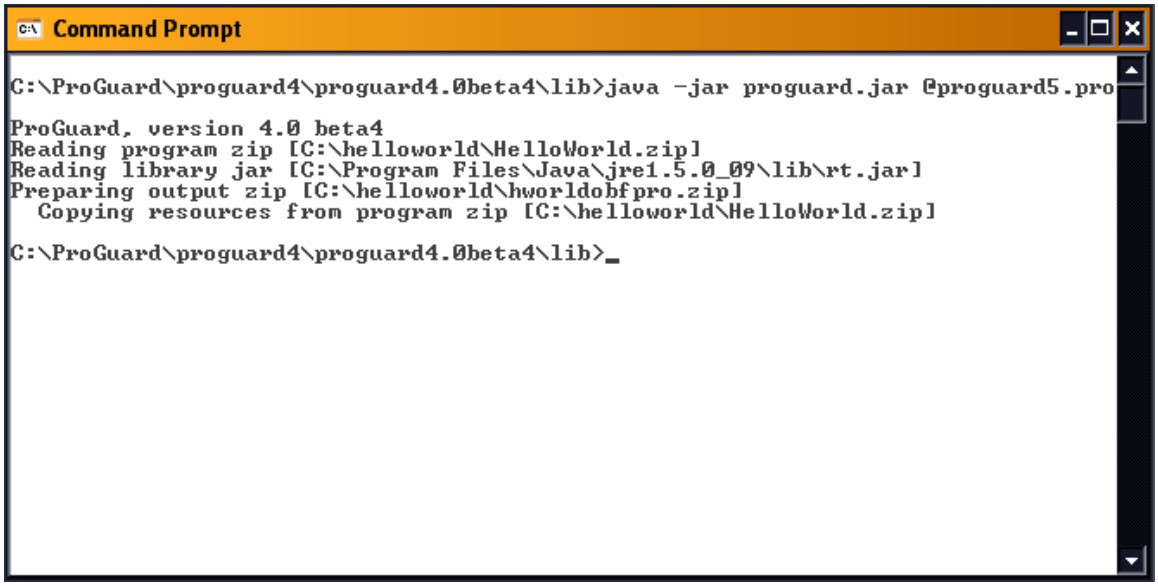

Fig B.1

The Fig. B.1 show the working of proguard jar file which is used to obfuscate the Java class files.

- @proguard.pro: is the configuration file which consists of input and output files which is used to obfuscate.
- This test is taking .zip file as input, this zip file consists of some class file which need to obfuscate, and the rt.jar is a jar(Java Archive) file where the result, the obfuscated class files, are kept.