THREATS AND TRUSTED COUNTERMEASURES USING A SECURITY PROTOCOL IN THE AGENT SPACE

Sayed Nouh
Egyptian Consultant, Egyptian Embassy

and

Tinbit Admassu
Department of Electrical and Computer Engineering

ABSTRACT

Mobile Agent computing is a paradigm of distributed computing, that has generated considerable excitement in the research community. Despite that, it has not been translated into a significant number of real-world applications due to a new dimensionality of security problem it brings along with it. In this paper familiarization to Mobile Agent technology and threat of hostile host towards a visiting agent is given due diligence: Malicious host problem. The threats are identified and a modified mobile computing model is proposed to prevent some of the threats. A prototype that realizes the concept is implemented using IBM's Mobile Agent platform, Aglets.

Keywords: *Agents, Malicious host problem, Mobile Agents, Trusted nodes.*

INTRODUCTION

A Mobile Agent can be thought of as a program, which can autonomously migrate between various nodes of a network and perform computation on behalf of a user [1]. It has a unique ability to transport itself from one system in a network to another. The ability to travel allows a Mobile Agent to move to a system that contains the object with which the agent wants to interact and then take advantage of being in the same host or network as the object. Mobile Agents are promising paradigms for the design and implementation of distributed applications [1].

Mobile Agent technology is not entirely based on Mobile Agents only, there is another complementary component called Mobile Agent platform. It provides appropriate execution environment and services to the Mobile Agents. The agent along with the Mobile Agent platform is called Mobile Agent System (MAS). Mobile Agent Systems can be roughly divided based on the programming language by which they are developed and use: Java and non Java based (using languages like C/C++ and scripting languages like Tc1/Tk). Around 80% of Mobile Agent systems available today are built using Java, due to its inherent support to Mobile Agent programming.

The rest of the paper is organized as follows. In section 2 the threats of hostile hosts towards a visiting agent is identified and some of the available countermeasures to those threats are presented. In section 3 the proposed system design is explained. Section 4 presents the capability and performance of the proposition using a prototype developed and the last section, section 5, draws a conclusion.

SECURITY IN THE AGENT SPACE

The security issues of MAS are of multidimensional. An agent could attack a platform, an agent could attack another agent and a platform could launch an attack against its visiting agents. The first two attacks have their counter part in the traditional client server environment. But the last kind of attack, a platform launches an attack against its visiting agent, is the most difficult of all attacks to solve. Some researchers even claim that it is impossible to solve. In this paper we will be looking into this attack (Malicious Host Problem).

Hostile host threats

These types of threats represent a class of threat, where the host compromises the agent. The hostile actions include: Masquerading, Denial of Service, Eavesdropping and Alteration of carried result. These attacks are most difficult to be detected and prevented, since the host has a full control of the agent's code and data.

Masquerading: An agent platform can masquerade as another agent platform in an attempt to deceive the Mobile Agent as to its true

destination. The problem has more to do with the capability of a visiting agent to correctly identify and authenticate its executing host, while it is actually on it. As an example, a Mobile Agent entrusted with the task of finding the "lowest price" of a commodity by visiting various virtual shops, can be tricked by a malicious masquerading platform, by making it believe that all other shops have quoted a higher price. Thus, the masquerading platform can harm both the visiting Agent and other Agent platforms [3, 4].

Denial of Service: When an agent arrives at an agent platform, it expects the platform to execute the agent's request faithfully and provide a fair allocation of resources. A malicious platform, however, may ignore agent's service requests, introduce unacceptable delays for critical task or even terminate the agent without notification. Agents, if any, which are waiting for a result from a non-responsive agent on malicious platform must be careful to avoid becoming deadlocked.

An agent can also become live locked if a malicious platform or programming error creates a situation in which some critical stage of the agent's task is unable to finish because more work is continuously created for it to do. Agent live lock differs from agent deadlock in that the live locked agent is not blocked or waiting for anything, but is continuously given task to perform and can never catch up or achieve its goal [3].

Eavesdropping: The classical eavesdropping threat involves the interception and monitoring of secret communications. The threat of eavesdropping, however, is further exacerbated in Mobile Agent systems because the agent platform can not only monitor communications, but also can monitor every instruction executed by the agent, all the data it brings to the platform, and all the subsequent data generated on the platform [3,5].

Alternation: Alteration threatens the integrity of the agent as a whole. As already discussed earlier when an agent arrives at a given host, it exposes its code, state and data to the platform. A mechanism that ensures the integrity of the agent needs to be in place [5].

Countermeasures for malicious host threats

Over years, a number of countermeasures for malicious behavior of hosts towards a visiting agent have been proposed, some of them are applicable, while others have only of a theoretical significance. Generally, the countermeasures provide either detection or prevention mechanism to the visiting agent.

Trusted Hardware: This countermeasure tries to enforce the notion of trust between an agent and a host by physically adding, secure – tamper detecting and responding hardware to conventional computing systems. The hardware encapsulates the entire environment in which the agent executes, creating a safe heaven within hosts in the agent space. It protects the visiting agent from any possible attack that could be launched by the entertaining host [6].

Trusted Execution Environment: This method is a variation of the above method. It eliminates the deployment of the specialized hardware. Instead, according to this method a set of trusted nodes needs to be setup in the agent space prior to any agent to host interactions [1].

Computing with Encrypted Functions: This method prohibits the executing host from learning anything substantial about the agent. It has been suggested by Sander and Tshudlin and tries to ensure the computation privacy of the agent in the untrusted host. Accordingly, functions will be encrypted such that their transformation can again be implemented as programs. The resulting program will consist of instructions that a processor understands, but the processor will not understand the "program's function" [4].

Code Obfuscation: Code obfuscation is suggested by Hohl [2]. According to this proposition, an algorithm called obfuscating algorithm will be used to mess up the code, while still creating a semantically equivalent version of a program. The idea is to make the program behave like a black box [6].

PROPOSED SYSTEM DESIGN

The proposed countermeasure is based on trust. Generally a trust that a Mobile Agent has on a particular host can be blind folded, based on policy enforcement or based on control and punishment.

A blind folded, kind of trust, is the one in which the Mobile Agent "simply need to trust its entertaining host". In this scenario, the host can do whatever it wants while giving services to the Mobile Agent, but still it is trusted that it neither has malicious behavior nor collaborate with other hostile hosts that perform some evil action on the agent.

The second kind of trust is based on policy enforcement. In this case, the Mobile Agent and the host have a prior contractual relationship in the form of policy. Such kind of trust should work fine as long as the signing parties conform to their rights and obligations.

The last kind of trust is based on control and punishment. Here no prior policy needs to be signed between the two parties. Although there is no contract signed it is not a blind folded trust as in the first case. The trust assumes that hosts are not by nature malicious and give them a chance to behave accordingly. But it still uses control mechanism to punish the host if found guilty of misbehaviors.

The proposed countermeasure uses a combination of the above two kinds of trusts, based on to which nodes, in the computer network, the Mobile Agent is interacting: based on policy enforcement and based on control and punishment. Before we move on to describe the proposed countermeasure, the next section outlines the guidelines used to develop the countermeasure.

Design guidelines

The following points are used as guidelines when the proposition is being developed:

- Convenience to the owner of the agent.
- Abstraction of the modification.
- No pre negotiation with hosts.
- Ease of access of information gathered.

The proposed countermeasure

A closer look and evaluation of the various kinds of attacks launched by hostile hosts' reveals that, Mobile Agents are subjected to such kind of attacks because they are a lonely figure once sent to the agent space. Hence the proposition modifies the computing model of the mobile computation in order to address hostile host threats.

Figure 1 shows an overview of the mobile computing model, in which the proposed countermeasure is taken into account, with a number of additional elements. Much like the case of trusted third parties a node is setup in the agent space to provide a different task to the Mobile Agent. In this setup it is mandatory that the home or owner of the Mobile Agent has a public-private key pair at its disposal, the public key is published to the world, so that the Mobile Agent could retrieve this key while it is visiting hosts. These keys are used by the security protocol to protect the confidentiality and the integrity of parts of the Mobile Agent.

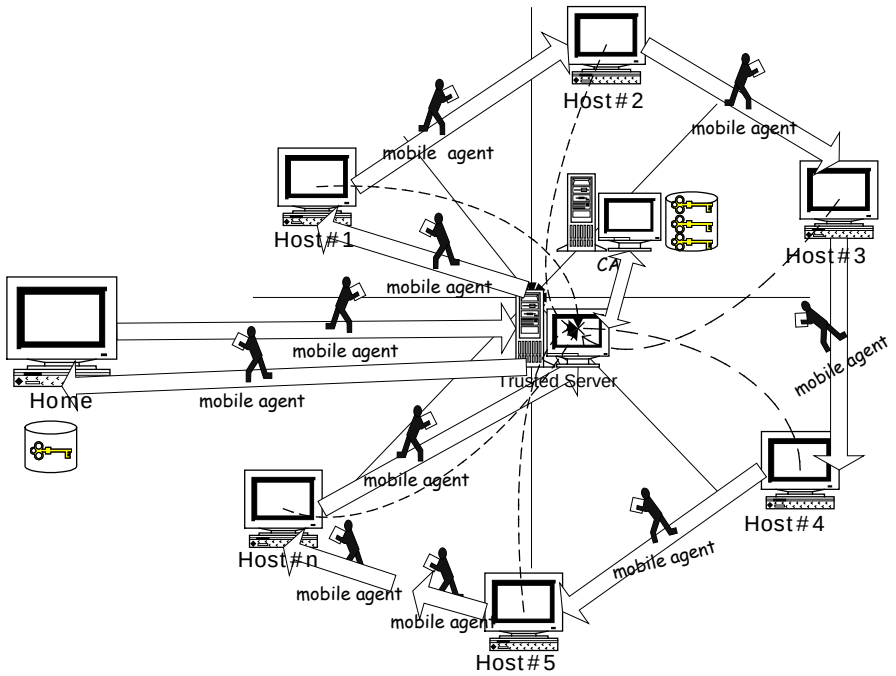

Figure 1 Proposed mobile computing model

As it can be seen from the Fig. 1, the proposal modifies the way by which the mobile computation is done. More specifically, the arrows dictate that, the Mobile Agent first goes to the trusted node, creates a temporary storage element called active storage element (ASE), then moves to the first host to be visited. It goes there, sends the information it has retrieved from the corresponding host to be stored temporarily at ASE. The trusted node accepts the information and stores it. Each Mobile Agent that has a trust relationship with this node does the same, creates its own ASE at the trusted node and uses it to store the partial information it retrieves from each hosts. At the end of its mission the Mobile Agent returns back to the trusted node and asks the corresponding ASE to hand it over the results it has been accumulating so far, carry back the result to its home as if it has been doing the job alone.

Unlike the original model of a unified agent space, it is assumed that the agent space is divided into regions, within each region a node called trusted server is setup. These servers provide various services to the Mobile Agent while the agent is in the agent space. The Mobile Agent supported by these third parties trusted nodes as well as the security protocol discussed later on should be able to avert some of the evil acts from hostile hosts. The division of the agent space into regions is analogous to the cells in the mobile communication systems. In case of mobile communication systems, at the center of each cell there is a transmitter and receiver, likewise in the proposed countermeasure at the center of each region there is a trusted server.

The concept of introducing a trusted server setup in a network handled by a third party is not new at all. If we look around a number of servers deployed in an internet work, they all or at least at some point in their operation provide synonym. Web servers, mail servers and root Domain Name Services (DNS) servers are some to mention. Let us take on the web servers as an example to highlight the similarity as well as the feasibility of the proposition.

These days we can develop our own web page and upload it to a web server for free. Let us take this argument a step further as it might seem less convincing in case of freely web hosting services. Take Ethiopian Telecommunication Corporation (ETC), it provides web hosting service with the amount of fee to be paid depending on the size of file we want to upload as well as other features our page requests from their web server. In either case all that is needed from our side is to pay the price. Indeed the two parties, the ETC and the one who wants to get hosting service, have to be agreed on terms of use. The ETC once agreed on to host our page, it provides all the necessary computational resources, when our page is being viewed by all around the world. The ETC has the obligation not to modify the content of our page without our permission, hence the ETC should display our page as it is. This has a strong implication on the feasibility of a trusted server set up in the agent space by a third party which could provide a processing service to the Mobile Agent without altering the data or code of the Mobile Agent. Much like the terms of use signed between the above two parties, here also terms of use could be signed between the user of the Mobile Agent (home node) and the trusted server in the agent space in a form of policy enforcement. Hence trusted servers will not modify the Mobile Agent's content, as web servers do not modify the web page they host. But here the security protocol provides further protection to the Mobile Agent content at the trusted server.

It is such a similar concept that the proposition wants to exploit. The nodes and the trusted servers could be set up, in a similar style as nodes of root Web servers, by the Mobile Agent user community. More specifically by the huge set of nodes that are set up to be visited by the Mobile Agent.

In section to follow, we will take a look at the main components of the proposal and how should the components interact according to the security protocol.

Components of the proposed countermeasure

Figure 1 depicting the overall view of the proposal shows that the countermeasure constitutes various components at various degree of multiplicity. Each of these components are listed and defined as follow:

(i) Home of the Mobile Agent,

(ii) Mobile Agent (MA),

(iii) Trusted Node (TS),

(iv) Active Storage Element (ASE) and

(v) Host.

Home of the Mobile Agent (Home):

It is the computer running Mobile Agent platform and has sent the Mobile Agent to carry out a task on its behalf. It can also be defined as a computer

running a Mobile Agent based distributed application. The application as a part of its mission packs a task into the Mobile Agent and sends it to the agent space. The Mobile Agent after completing its task will eventually return to the home carrying the result.

Mobile Agent:

As defined throughout this paper, it is a program that migrates from one node to another node in a computer network to accomplish a task given to it by its owner.

Trusted Node:

It is similar in composition to the home of the Mobile Agent, but differs in the function it provides. It is there to provide support and service to the Mobile Agent while it is in the agent space.

Active Storage Element:

It is a temporary storage element that is created by each Mobile Agent, at the trusted server, that is sent to visit nodes in the agent space. It actively participates in the process of temporary information storage and handing over of all the information to the Mobile Agent.

Host:

It is a computer in the agent space running Mobile Agent platform and entertains any visiting Mobile Agent which would like to gather information from it. This component is at the center of the controversy, which could be hostile. The host provides all the necessary resources for the agent to execute there.

Security protocol

A security protocol that defines how the basic components of the system (Home, Trusted Nodes and Active Storage Element) should interact with each other as well as what are the needed tasks to be performed at each level, so as the whole system could stand against the possible hostile host threats, is developed.

The security protocol alters what the Mobile Agent constitutes depending on where it is. While the Mobile Agent is transiting between its home and trusted nodes the usual composition is deemed. But when the agent is in the agent space visiting different nodes it has assumed to be composed of only the two out of the three components that is usually associated with: code and state, to give hostile hosts no chance of disclosure of information collected from previous hosts.

The security protocol also develops a mechanism that lets the user of the Mobile Agent to digitally sign the list of destinations it wants the Mobile Agent to visit. After forming a destination object which contains the list of all hosts the Mobile Agent is going to visit, it digitally signs the destination object using its private key, then the destination object is passed down to the Mobile Agent. The Mobile Agent upon its arrival at each and every host in the agent space verifies that it has a valid copy of the destination object before putting that object into use. So the Mobile Agent avoids the possibility that it would be directed to visit other hosts by altering the list of paths it has carried from its home, as any malicious host could not counterfeit the digitally signed destination object. Let us see, step by step, the security protocol in action and its effects on the components of the Mobile Agent system. It is assumed that, the home node has a public-private key pair (HPubK-HPrvK).The public key could be retrieved by the hosts from relevant authorities.

At Home, as shown in Fig. 2**:**

- The user of the MA specifies the address of the list of hosts it wants to be visited using the Agent Based Application (ABA),
- The ABA accepts the list and forms a destination object. The destination object includes the list of hosts to be visited, the address of the trusted server (TS) and the home,
- The ABA digitally signs the destination object and passes it to the MA which is programmed to perform the required task and
- The MA accepts the signed object. By using HPubK, the MA verifies that it has the right unsigned destination object from which the address of the next node to be visited is determined and dispatches itself to that node, as pointed out in the previous section it goes first to the Trusted Node.

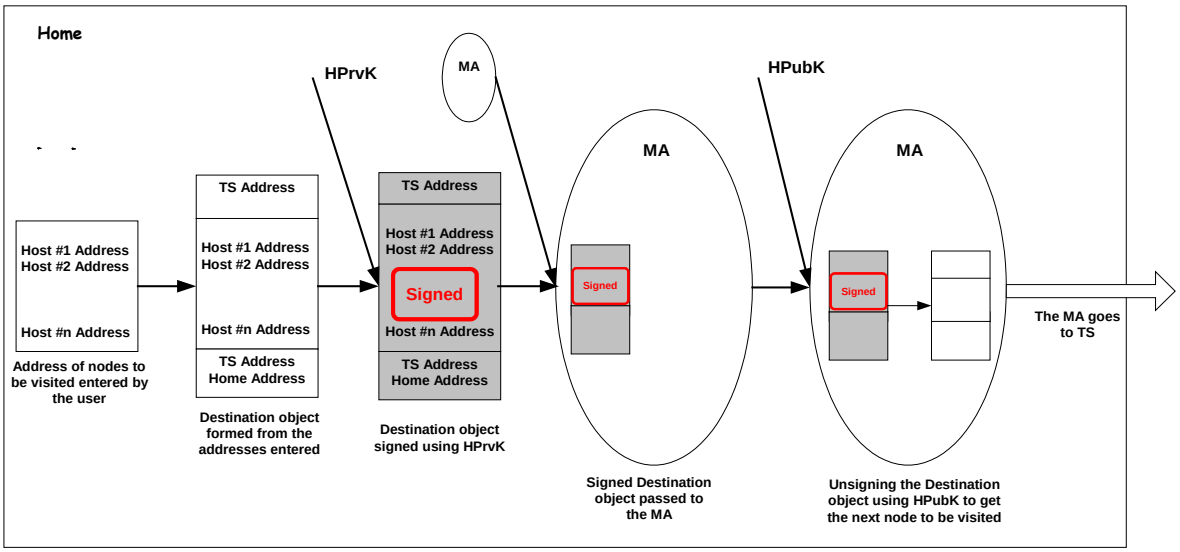

Figure 2 Security protocol at home

At Trusted Server (TS), as shown in Fig. 3:

- The MA arrives at the TS, creates its own Active Storage Element (ASE),
- The MA passes down the necessary information to the ASE so it can effectively communicate with it,
- The MA retrieves the public key of the home, HPubK,
- Using this key, the MA unsigns the digitally signed destination object and determines the next node to be visited. In this case it is the first host in the list and
- The MA Dispatches itself to that node.

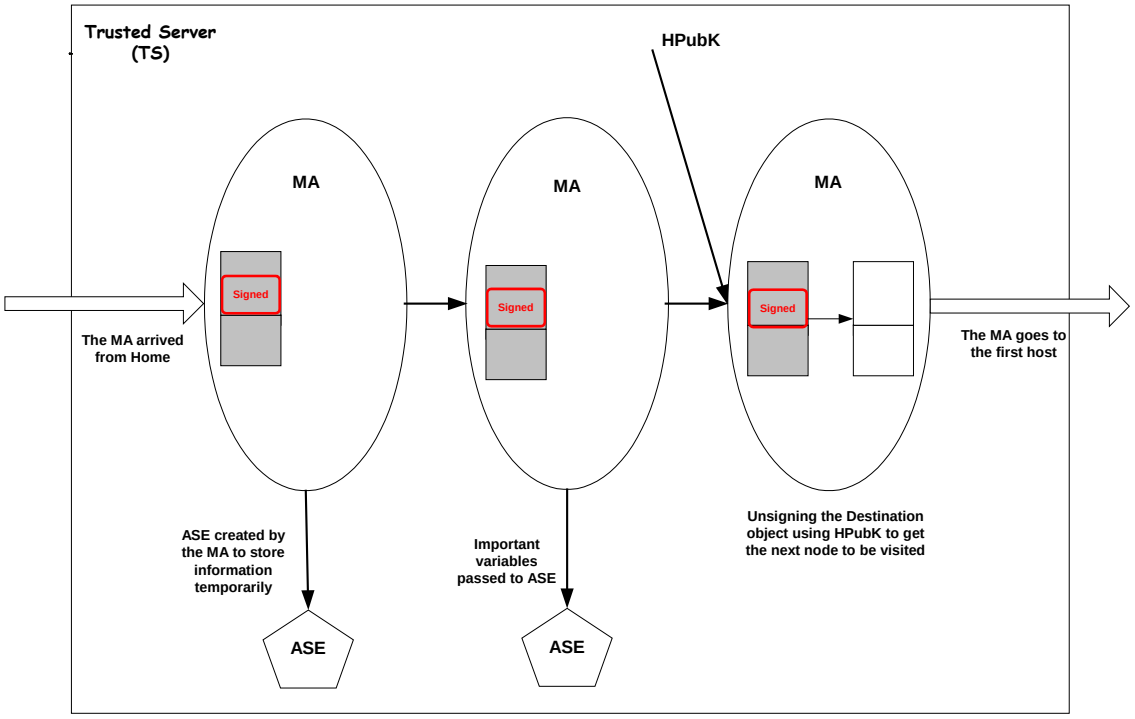

Figure 3 Security protocol at TS

At ith host, as shown in Fig. 4:

- The MA arrives at the ith host, Host_i,
- It generates a random symmetric key, SymK_i,
- The MA retrieves the public key of the home, HPubK,
- Asks the host about the information it wants, Info_i,
- Encrypts the information using the symmetric, SymK_i(Info_i),
- Encrypts the randomly generated symmetric key using the public key, HPubK(SymK_i),
- Sends both of these information, HPubK(SymK_i) and SymK_i(Info_i), to its ASE at the TS to be stored temporarily,
- The MA unsigns its destination object, looks the address of the next node to be visited and dispatches itself to that node.

If the next node is another host it does the same task as indicated above. Else if it is a trusted server, the following set of actions follows.

At Trusted Server, as show in Fig. 5:

- The MA arrives at the TS, in its last leg of journey,
- Asks the corresponding ASE to hand it the overall information it has been accumulating so far (a pair of HPubK(SymK_i) and SymK_i(Info_i) retrieved from each host), and takes these information,
- After unsigning its destination object, it looks for the address of the next node to be visited. In this case for sure it is the home node and
- The MA dispatches itself to its home.

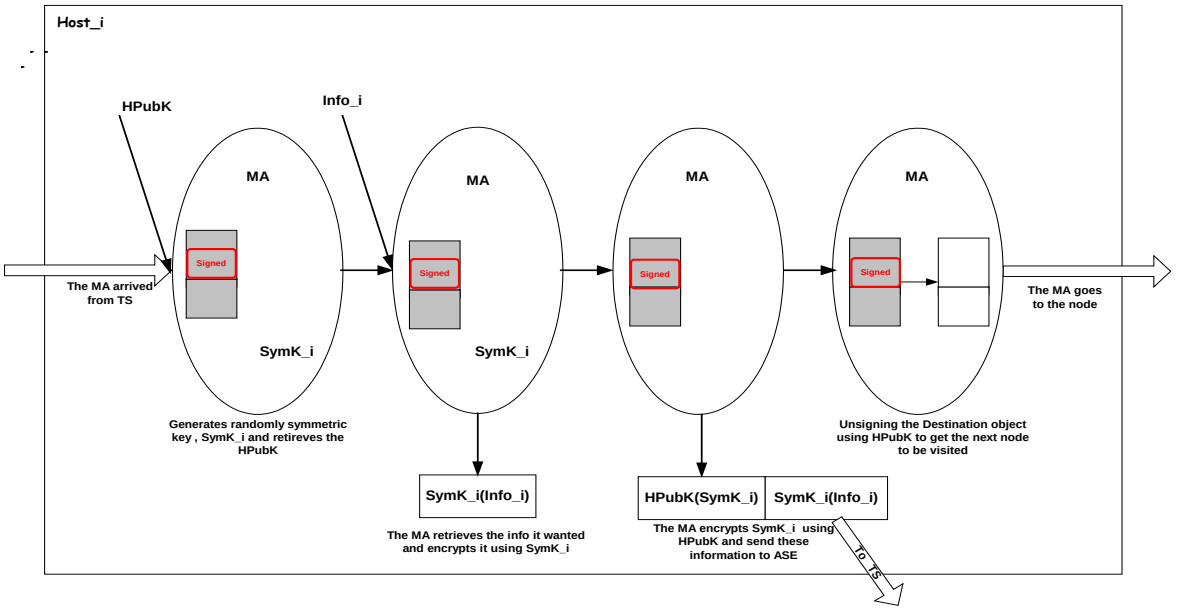

Figure 4 Security protocol at the ith host.

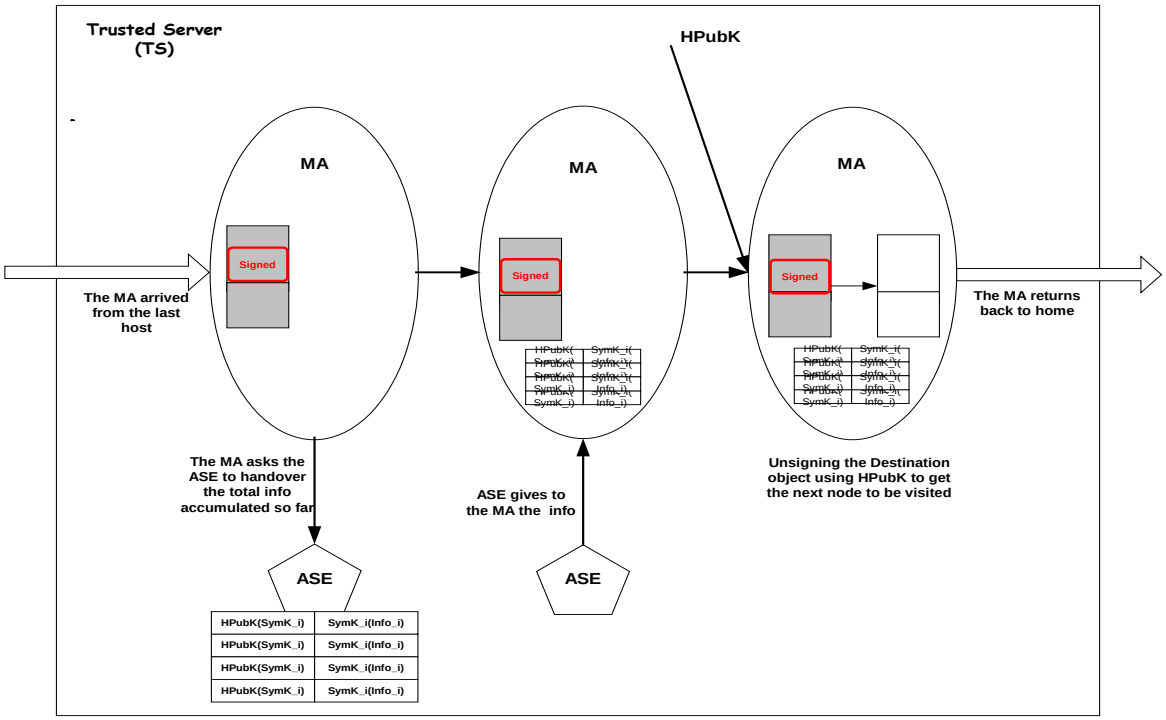

Figure 5 Security protocol back at TS

At Home, as shown in Fig. 6:

- The MA arrives back at home after doing the task assigned to it,
- The MA contains a pair of encrypted information,
- The MA hands the overall information to the ABA. Note that they are all in encrypted form.
- For each pair of encrypted information retrieved from each host, the ABA does the following:

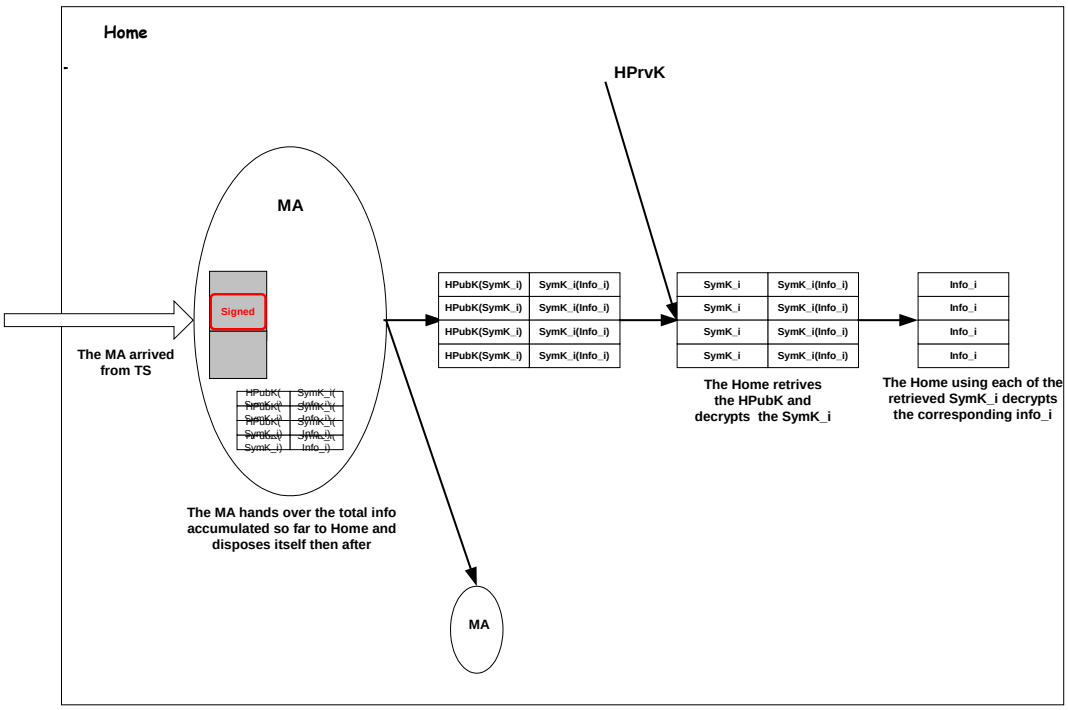

Figure 6 Security protocol back at home

- First, using its private key (HPrvK) to decrypt the encrypted symmetric key, HPubK (SymK_i) ,
- Second, using the decrypted symmetric key, SymK_i, it decrypts the information which is encrypted using the same key, SymK_i (Info_i),

- The ABA does the same process for each pair of information retrieved from every host the agent goes to collect information and
- At last the ABA displays the result to the user, Info_i.

Security Protocol Summary:

- For N hosts:
  - N hosts addresses digitally signed by the home node,
  - One ASE created at TS,
  - N symmetric random keys generated at each host,
  - N information retrieved will be encrypted by the corresponding N symmetric keys,
  - The N symmetric keys will be encrypted by the public key (RSA) of the home,
  - The encrypted N information and encrypted keys stored at the ASE,
  - Decryption at home node and
  - Displaying the plain text result to the user.

RESULTS AND PERFORMANCE COMPARISON

To access the capability and the cost of the proposed countermeasure, the following techniques and equipments are used: Two personal computers with 768MB and 512MB of RAM respectively, which run Windows Server 2003 and a number of Mobile Agents (Proposed MA, DS MA and Normal MA) as described below, are used.

Proposed Mobile Agent: Proposed MA

A Mobile Agent which is governed by the security protocol and hence performs mobile computation as pointed out in Section 4.

Digitally Signed Mobile Agent: DSMA

A Mobile Agent that supports digital signing of the destination object while still performing computation in a way discussed in Section 3.4.

Normal Mobile Agent: Normal MA

A Mobile Agent that performs mobile computation in the usual way.

Each of the above Mobile Agents (Proposed MA, Normal MA and DS MA) are given a similar task to carry out.

Results:

To measure the capability of the proposal towards eavesdropping threat, a test environment is set up using the above mentioned computers as shown in Fig. 7. Computer A takes up the position of trusted server (TS) and computer B runs many host nodes simulated through various port numbers as well as the home node. Ethereal Network Packet Analyzer software is run on computer A. It is open source freely available packet analyzer software that can capture and store packets from a live network for further processing. This packet analyzer software is made to sniff into packets exchanged between the two computers as the various types of MA's do their job. The Fig. 8 shows analysis of the packet captured while the Normal MA and DS MA are in operation.

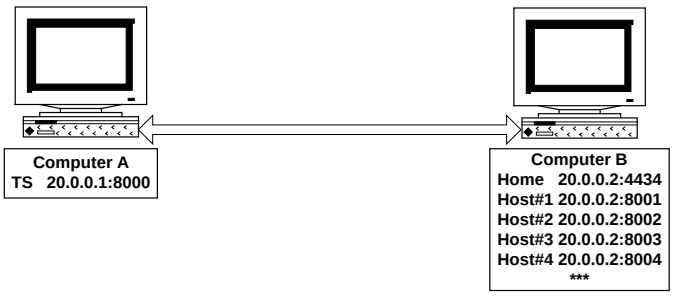

Figure 7 Test environment set up

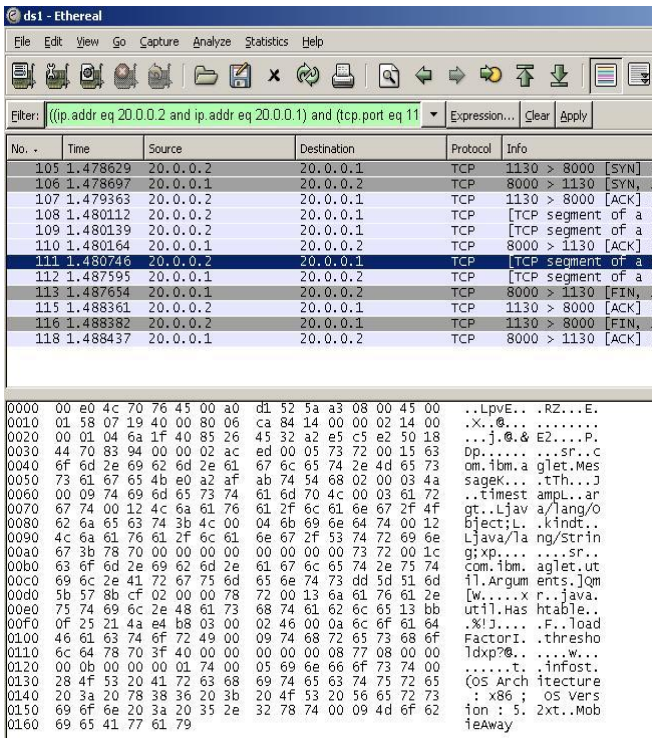

Figure 8 Captured packet analysis for DS MA and normal MA.

As it can be seen from Fig. 8, in either of the cases it is possible to eavesdrop what information is retrieved and exchanged at each host: "OS Architecture: x86; OS Version: 5.2" seen at the bottom right corner of the window.

Figure 9 shows analysis of the captured packet while the Proposed MA is in operation. As it can be seen from the figure, unlike the above case since the information is sent to the TS in encrypted form, it is not possible to look into its content. Hence the security protocol provides the required confidentiality of the information while it is being stored at ASE.

To test the capability of the proposed countermeasure, towards Alteration threat, a similar test environment as in the above case is used, except that all of the nodes are simulated in computer.

A hostile node is introduced on a different port number in the same computer, Computer B. This node is planned to behave maliciously towards the Proposed MA. Specifically, it is planned to supply a wrong public key to the MA as the MA arrives there and is in the process of unsigning its digitally signed destination object. But fortunately the MA cannot unsign the signed object using the public key just supplied. This is because the destination object is signed by the private key of the home node, not by a private key which corresponds to the public key supplied by the hostile node. Hence any attempt of alteration of destination object will be detected by the MA.

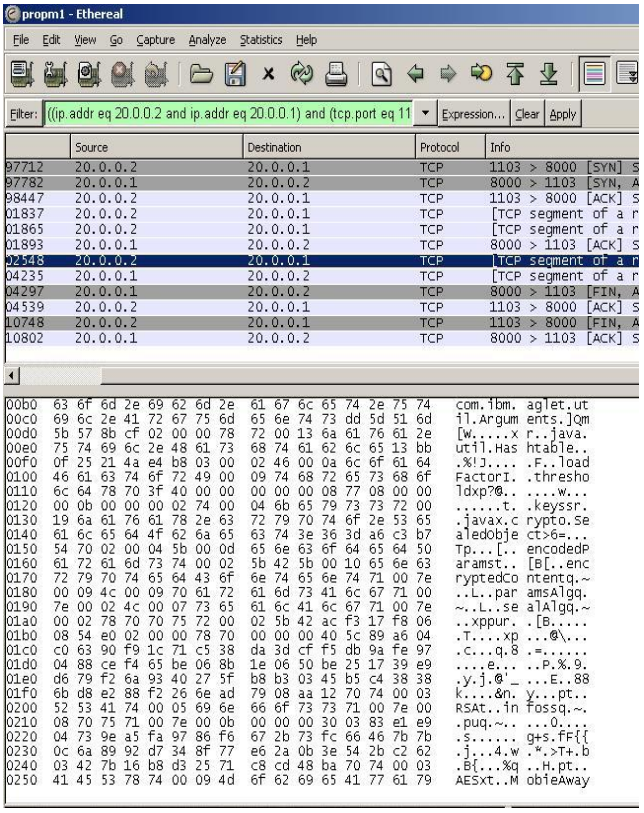

Figure 9 Captured packet analysis for Proposed MA

Performance Comparison:

To measure the cost of the proposal, a similar test environment as above is used. Each of the above Mobile Agents (Proposed MA, Normal MA and DS MA) are given a similar task to carry out. Their performance is compared in terms of their average turn around time, measured in milliseconds (ms).

This performance parameter is the average time in milliseconds (ms) each Mobile Agent requires to do the job, after dispatched till it returns and handovers the result to the user. Figure 10 indicates just that for the three scenarios (Proposed, Normal and DS). As might be expected DS Mobile Agent takes in between of the two. Comparing the execution time of the Normal MA with the Proposed MA, the Proposed MA needs approximately 4x more time. This substantial amount of time is a price to pay to achieve the corresponding security. Generation of the keys,

encryption of partial information, verification of destination object at each visited host and at last collecting the results back to the Mobile Agent from the TS all add up to form a big turn around time.

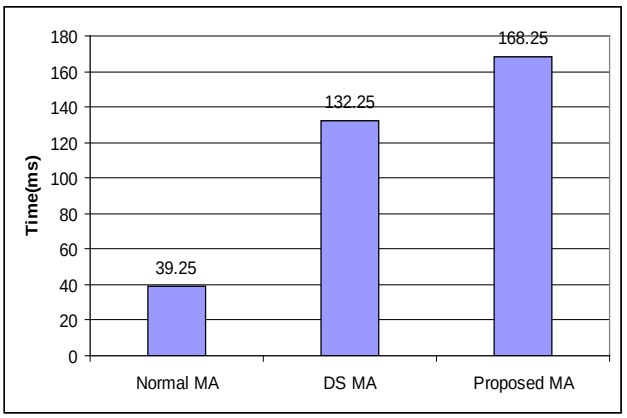

Figure 10 Actual performance time comparisons between the three scenarios

Comparing the performance time between DS MA and Proposed MA, the DS MA needs less time. This is due to the fact that it (DS MA) does not carry out some of the functions the Proposed MA performs like: Generation of keys, Encryption and others. It takes time only as it verifies that the destination object is valid copy on its arrival at each and every host.

Figure 11 compares the trend of the execution time for all Mobile Agent cases. As the number of nodes to be visited is steadily increased, we notice that the turn around time increases. This is tribute to the fact that there are more jobs to be done.

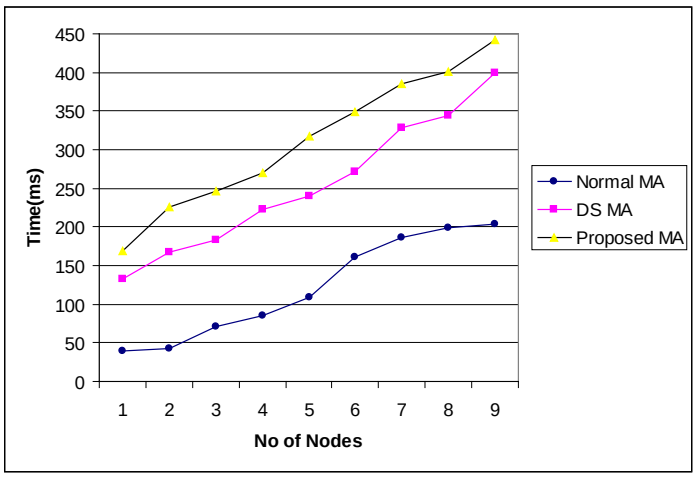

Figure 11 Performance time trend as the number of nodes visited increases

CONCLUSION AND FUTURE WORK

In this paper we have tried to address the issue of Mobile Agent security at the same time providing familiarization to the concept of Mobile Agents programming. The paper more specifically was an attempt to try to look in to a notoriously difficult task as pointed out by many researchers. An attempt has been made to avert some of the malicious host's threats by adopting a number of mechanisms to the way the original computation is made by the Mobile Agent. The counter measure proposed introduces the concept of setting up another home in the agent space as called "home away from home" for partial result storage and the separation and digital signing of the destination of the Mobile Agent.

REFERENCES

[1] Rahula Jha, *Mobile Agents for e-commerce*, M. Tech. Dissertation, IIT Bombay, India, 2002.

[2] Lange, D. B. *Mobile Objects and Mobile Agents: The Future of Distributed Computing*, Springer-Verlag Berlin Heidelberg 1998.

[3] Jansen W. and Karygiannis T. *Mobile Agent Security*, National Institute of Standards and Technology, Gaithersburg, MD 220899.

[4] Vijil E. C. *Secuirty Issues in Mobile Agents*, M. Tech. Dissertation,IIT Bombay, India, 2002.

[5] Altalayleh, M. and Brankovic, L. *An Overview of Security Issues and Techniques in Mobile Agents*, University of NewCastle, Australia.

[6] Bierman, E. and Cloete, E. *Classification of Malicious Hosts Threats in Mobile Agent Computing*, Proceedings of SAICSIT 2002, Pages 141-148, University of South Africa.