

A Model for Mobile Agent Security in E-Business Applications

A. Kannammal

*Department of Computer Applications
Coimbatore Institute of Technology,
Coimbatore 641 014, TamilNadu, INDIA
kannaphd@yahoo.co.in*

N.Ch.S.N. Iyengar

*School of Computing Sciences
Vellore Institute of Technology University,
Vellore-632014, Tamilnadu, INDIA
nchsniyr@gmail.com*

ABSTRACT

Mobile agent systems provide a great flexibility and customizability to distributed applications like e-business and information retrieval in the current scenario. Security is a crucial concern for such systems, especially when they are used to deal with money transactions. Mobile agents moving around the network are not safe because the remote hosts that accommodate the agents can initiate all kinds of attacks and can attempt to analyze the agents' decision logic and their accumulated data. Hence, mobile agent security is one of the most challenging unsolved problems. This paper analyzes the security attacks on mobile agents by malicious hosts and seeks to address some of these problems by proposing solutions based on public key authentication techniques and cryptography. The authors develop an experimental application and evaluate the security and performance of proposed solutions. They develop a performance model in order to tune the parameters of execution environment to meet the desired level of performance and security.

Keywords: Mobile agent, e-business, security, performance model

1. INTRODUCTION

Mobile agents are autonomous software agents that travel in a computer network to execute and perform tasks on different hosts on behalf of their owners. Autonomous mobile agents bring advantages such as task delegation, network communication, and cost reduction for distributed tasks [2]. Security is one of the blocking factors for the development of these agent-based systems [12].

The problem of mobile agent security can be divided into two parts: (1) the protection of hosts against agents, and (2) the protection of agents against hosts. The first part – host security problems – can be effectively solved by strongly authenticating the code sources, verifying code integrity, and limiting access rights. This solution is realized in the Java security model [14]. The second part – agent security problems – is the main unsolved issue for mobile agents because of the possible existence of malicious hosts that can manipulate the execution and data of agents [3]. The lack of a trustworthy computing base [15] can add new complexities to the problem. The security issues related to these agent-based systems must be addressed in order to exploit the potential benefits offered by mobile agents. The main objective of this paper is to propose a solution to mobile agent security problems, and to design the solution and analyze it.

To experiment with mobile agents, the authors build a mobile agent system, the *Shopping Consultant Agent System (SCAS)*, using the Java Agent Development Environment (JADE) [4]. The SCAS is useful in collecting the prices of a set of products specified by users from different seller hosts in an electronic market. Next, the authors address security issues for the system, evaluate possible attacks by malicious hosts, and devise and implement solutions to protect the system against these attacks.

Section 2 discusses mobile agent security problems, reviews existing techniques to protect agents, and presents various methods that are proposed to address agent security issues. Section 3 describes an experimental system to identify the security problems of agents in a real-time application and offers a proposed solution. Section 4 describes implementation of the proposed solution, compares it with the existing model, and develops a performance model to strengthen the experimental evaluation. Section 5 offers future research directions.

2. SECURITY ISSUES AND EXISTING SOLUTIONS

Like any distributed system, the mobile agent system is subject to security threats such as eavesdropping, corruption, masquerading, denial of service, replaying, and repudiation. Issues such as encryption, authorization, authentication, and non-repudiation, therefore, must be addressed in a mobile agent system. Moreover, a secure mobile agent system must protect both the hosts and the agents from tampering by malicious parties.

When an agent executes on a remote host, the host is likely to have access to all the data and code carried by the agent. If the host is malicious and abuses the

code or data of the agent, the privacy and secrecy of that agent and its owner would be at risk. Various types of attacks may be carried out by malicious hosts [3], such as spying out and manipulating code, data, and control flow, incorrectly executing, masquerading, and returning wrong results of system calls to agents. A number of solutions have been proposed to protect agents against malicious hosts [16]. These can be divided into three types: *closed network*, *agent-tampering detection*, and *agent-tampering prevention*. None, however, solve the problem completely.

2.1. Existing Solutions to Protect Mobile Agents

Two of the proposed solutions that look most feasible and interesting to protect mobile agents are:

Protected Agent States [10], which are basically signing and encrypting of agent states based on public key cryptography that helps to achieve the confidentiality and integrity of the agent's data. They are effective and feasible in this regard because of the well-established cryptography theory underneath. However, they do not protect the code integrity and confidentiality of agents.

Mobile Cryptography [15], which is a possible approach to protecting agent code integrity, works as follows: *If Alice wants Bob to evaluate a function f for her, based on Bob's data x , she would encrypt the function f to produce $E(f)$, and implement a program P that evaluates the encrypted function $E(f)$, and sends P to Bob. When Bob runs P , he would not produce plaintext output $f(x)$ that he can read and modify. Instead, he can produce only the encrypted output $E(f(x))$, which would be readable only by Alice, who has the key to decrypt. Moreover, Bob cannot modify the execution of P , because it implements an encrypted function that Bob would not understand.* This approach is proved to be possible for polynomial functions only. However, if this approach really can be extended to other functions, such that arbitrary functions can have an encrypted but executable form that can be evaluated on a remote host, the problem of malicious hosts would be effectively solved.

Xudong [18] has proposed a Police Office Model, which separates the safety critical paths of mobile agents from malicious hosts. A hybrid method that mixes a function composition technique and a homomorphic encryption scheme is proposed in [7], to prevent only privacy attacks and integrity attacks on agents. This approach is an extension of mobile cryptography.

Page [11] has demonstrated the applicability and the effectiveness of the Buddy model of security for the agent community in different pervasive scenarios. This approach is suitable for a pervasive computing environment. Mobile agent security is also enhanced by integrating a trusted platform module with the mobile agent platform itself [17]. The same approach is taken in [6] to develop a Java-based mobile agent infrastructure so that this infrastructure can be adopted for distributed applications such as network management in spite of the

security concerns of mobile agents. These two approaches are based on the idea of improving the mobile agent platform itself.

A fair and secure mobile agent environment based on a blind signature and a proxy host is proposed in [9], which tracks malicious mobile agents only after a service host is attacked, and which protects the manipulation of information carried by the mobile agent. Our detailed study of existing techniques reveals that no single existing solution can currently protect mobile agents from all the basic security vulnerabilities explained later, in section 3.1.

2.2. Existing System Model: SACS

The Shopping Consultant Agent System (SCAS) is a web-based mobile agent system that provides users with information on the products in an electronic marketplace. It is written in the Java programming language, on top of the Java Agent Development Environment (JADE) [4].

SCAS enables mobile agents to retrieve product information in an *electronic market* for users. An electronic market consists of hosts that sell products on the network. Each seller maintains a database of details of products available at that host. SCAS allows users to specify a set of products and the corresponding quantities they want to buy. An agent is created for the user, which will collect price details from hosts in the e-marketplace. The itinerary of the agent is determined before the agent is launched. After the agent visits all hosts specified in its itinerary, it returns to its sender and reports the prices. The architectural components of the existing system model are depicted in Figure 1.

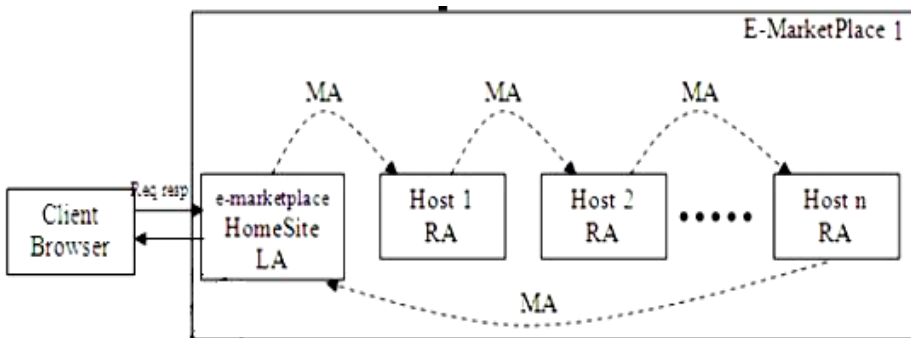


Figure 1. Architectural Components of Existing System Model (SACS)

As indicated, the major components of SACS are:

- **Client Browser**, which initiates the action by sending requests for prices of required items by specifying product IDs and quantities
- **LauncherAgent (LA)**, which resides in the e-marketplace home site, is responsible for creating a mobile agent on behalf of the user, providing necessary details to the mobile agent, sending it to the

hosts, receiving it, and presenting the information as a response to the user.

- **MobileAgent** (MA) keeps a list of product IDs and a list of the corresponding quantities specified by users. It travels around the network and collects price information from different hosts.
- **RemoteAgent** (RA), which resides in each host, provides the price details to the mobile agent, by querying the corresponding database present in the host.

3. PROPOSED SECURITY DESIGNS OF SACS

System security is of crucial importance to applications in an electronic marketplace, where money transactions are involved. Both host security and agent security would be issues of SCAS. However, since SCAS is built using the Java, which provides strong security mechanisms [14] to protect hosts against malicious programs or agents through the use of Java Virtual Machine (JVM) and sandbox, the host security problem is very much simplified and solved. On the other hand, the agent security problem is still of much concern.

3.1. Security Problems

The four primary security requirements for SCAS are: *Integrity*, *Confidentiality*, *Authenticity*, and *Non-Repudiation*. *Integrity* requires that the information is not tampered with nor manipulated by unauthorized means [1]. *Confidentiality* requires that only the intended recipients are able to view the information. *Authenticity* requires that the identity of the entity can be easily verified so that access can be provided to the information. *Non-repudiation* requires that the sender of the information cannot deny the information sent at later stage.

Four possible types of attacks to agents can compromise the security of the system. When an agent goes to a malicious host, the malicious host can change (1) the product list, (2) the quantities of products the agent wants to query, and (3) the results that the agent has collected from previous hosts, since all these items of information are in plain text form. All of these actions by a malicious host violate the integrity, authenticity, and confidentiality of queries. A fourth type of attack would occur if the user selects a host based on the information provided by the agent and if the host later denies that it has not provided that information or price. In the latter case, all the effort made up to that point would become a waste of time and money. This type of attack violates the non-repudiation requirement.

3.2. Solutions to the Problems

At present, there is no single ideal solution to mobile agent security in general [5], which would protect the system against the four vulnerabilities described above. We, therefore, devise a simple approach to protect agents in SCAS against attacks from malicious hosts, based on protected agent states. It is actually a

hybrid approach blending existing solutions; namely, a closed network and agent tampering prevention.

Closed Network. The object, namely KeyServer, is introduced, which provides a public key infrastructure for agents and hosts in the system. It stores the public key of participating hosts and mobile agents. It also provides the public key details to registered LauncherAgent and RemoteAgents. This, in effect, establishes a closed set of hosts registered and known to the KeyServer. The architecture of the system with Keyserver as a component is shown in Figure 2. The detailed design of components is shown in Figure 3.

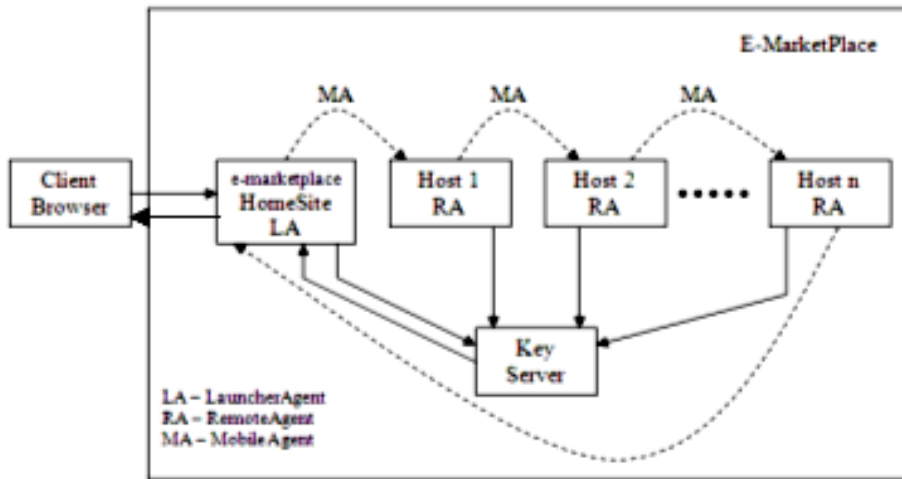


Figure 2. Architecture of SACS with KeyServer as Component

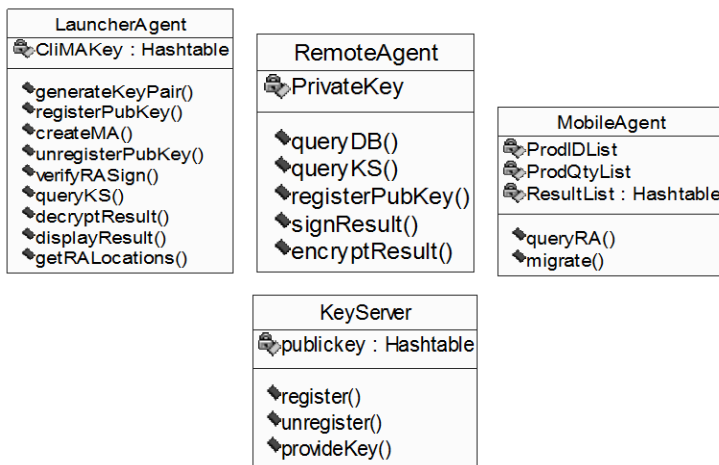


Figure 4. Detailed Design of Components with Attributes and Methods

Agent Tampering Prevention. To protect query integrity, an agent can digitally sign its list of products and quantities using its private key, before it is launched. A host receiving the agent should verify the product and quantity lists with the signatures by retrieving the public key of the agent from the KeyServer. Since only the LaunchServer possesses the private key for the agent, malicious hosts would not be able to fake the signature of the product and quantity lists. Moreover, each host should encrypt the query results returned to the agent with the public key of the agent. Therefore, only the LaunchServer can decrypt the query result, and confidentiality of query results is achieved. Furthermore, each host should digitally sign the query result it provides to the agent to ensure integrity, non-repudiation, and authenticity of the query result returned. The encryption algorithm chosen is the most common RSA algorithm [13].

The detailed workflow model of the system with proposed security solutions is given on the following page in Figure 4, which is self-explanatory.

3.3. Experimental Setting

JADE [4] is chosen as the implementation technology mainly because of its features and simplicity. It complies with standards like FIPA and MASIF. The system is developed using two Pentium IV systems, each with 128MB RAM with Windows as the operating system, and with both systems connected through 100Mbps Ethernet LAN.

4. EVALUATION

There are two aspects to evaluate the security design implemented. First, we analyze the security provided to SCAS by the additional measures. Second, we measure the performance overhead introduced to the system by such measures, according to different number of hosts in the system.

4.1. Security Analysis

The security of the additional measures lies mainly in the introduction of a key server. Assuming the key server and the communication channel with it are secure enough, which can be justified by the popularity of Kerberos [8] and Secure Socket Layer, the required closed network can be built effectively.

Preventing modification of the signed product and quantity lists by a malicious host is supported by the security of the RSA encryption algorithm [13]. The time complexity for breaking the system depends on the length of the key in number of bits. In the implementation, a key length of 1024 bits is chosen. This would be sufficiently secure for domestic purposes. Similarly, a malicious host would understand or modify the encrypted query results at the same complexity. Therefore, integrity of queries and confidentiality and integrity of query results can be achieved by prevention of tampering. The host provides details to the mobile agent after signing the information using its own private key. Also, the LauncherAgent verifies the signature before processing the results. Hence, the authenticity and non-repudiation requirements are met.

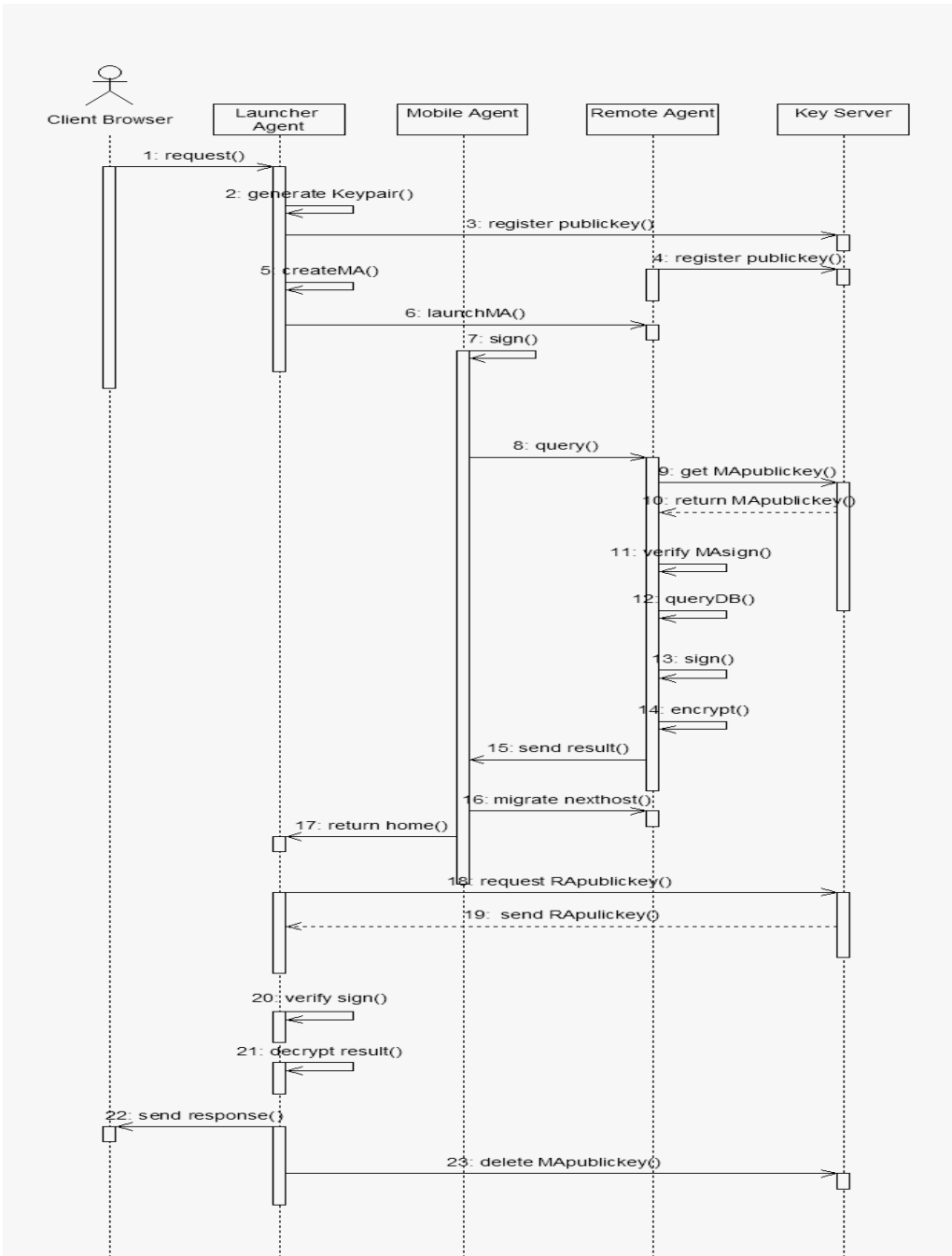


Figure 4. Detailed Workflow Model of SACS with Proposed Security Solutions

4.2. Performance Analysis

To evaluate the performance of SCAS against the scale of the system, we measure the times required for an agent to travel around an electronic market consisting of different number of hosts without and with security enhancements. The results are presented in Table 1 and are plotted Figure 4 and Figure 5. The data show that the Round Trip Time for an agent to travel in SCAS changes more or less linearly over the number of hosts in the system, both the cases. This is due to the additional time to travel an additional host. The overhead for each additional host is more or less the same.

The overhead introduced is due to the extensive use of the RSA algorithm to encrypt and decrypt each item, which is time consuming especially when the key is long. But a longer key gives a stronger protection to the system. Hence, a trade-off between performance and security for SCAS is identified.

Table 1
Average RTT for a Mobile Agent in SCAS

Number of Hosts	RTT Without Security Enhancements	RTT with Security Enhancements
1	3238.857	6921.18
2	5623.75	17267.64
3	8008.643	26440.18
4	10632.54	34036.23
5	13209.43	42982.81
6	16054.78	49830.77

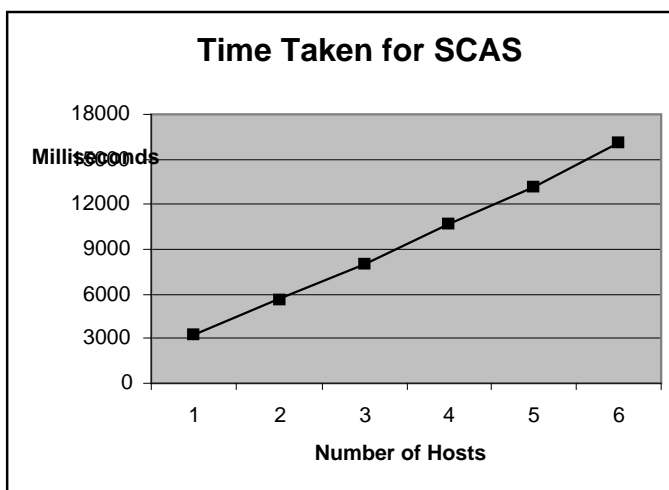


Figure 5. Average Turnaround Times Without Security Enhancement

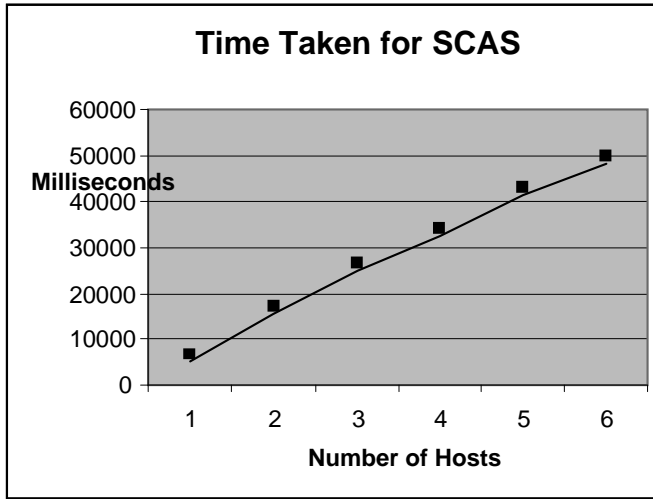


Figure 6. Average Turnaround Times With Security Enhancements

4.3. Performance Model for Secured SCAS

A performance model has been developed for the secured SCAS in order to evaluate the impact of proposed security solutions on the performance. The parameters selected for this case study, summarized in Table 2, are based on the workflow given in the previous section.

Since the MobileAgent does not perform the encryption task before it is launched, it does not generate the result of the task. It consists of code and state information. Its size D_{init} is given by:

$$D_{init} = D_{code} + D_{state} \quad (4.1)$$

MobileAgent signs the product id and quantity specified (the details of query). Hence its initial time for migration T_{init} is:

$$T_{init} = D_{state} * (1 / R_s) \quad (4.2)$$

When a mobile agent migrates among nodes, it performs its task and generates the result of the task. While a mobile agent is migrating among nodes, its size is defined as

$$D_{mig} = D_{code} + D_{state} + D_{data} \quad (4.3)$$

Table 2
Parameters for Evaluating Security Enhancements

Name	Description
N	Number of Nodes
D_{code}	Code size of MobileAgent (MA)
D_{state}	Status data size of MA (prod id and qty and data other than that collected from remote hosts)
D_{data}	Amount of data collected from remote hosts
R_g	Rate of key pair generation
T_{reqPK}	Time to request public key
T_{respPK}	Response time of public key
R_s	Rate of signature creation
R_v	Rate of verification of signature
R_e	Rate of encryption
R_d	Rate of decryption
R_{se}	Rate of Database search
R_{th}	Network throughput
$T_{reqdata}$	Time taken to request data
$T_{respdata}$	Time taken to respond to request data
D_{init}	Size of MA before migrating
T_{init}	Initial time required for MA migration
D_{mig}	Size of MA while migrating
T_{mig}	Time taken for MA to migrate and retrieve information
T_{Math}	Total time for MA migration (MA throughput)
T_{MAAuth}	Time taken for MA authentication
T_{Sdata}	Time taken for signing and encrypting the result
T_{MASec}	Time to provide security services
T_{SMA}	Total Execution time for Secured SCAS

The total execution time of the model amounts to the time for the information retrieval and the time for security services such as authentication, integrity, and confidentiality. Let T_{mig} be the time required that a MobileAgent moves to a remote host and retrieves the information. T_{mig} is given by:

$$T_{mig} = (1 / R_{se}) + (D_{mig} / R_{th}) \quad (4.4)$$

The total time required for a MobileAgent to execute the assigned task, T_{MAth} , is given by:

$$\begin{aligned} T_{MAth} &= (D_{init} / R_{th}) + T_{init} \\ &+ (D_{data} / R_{th}) + (N - 1) * T_{mig} \end{aligned} \quad (4.5)$$

The remote host has to authenticate the MobileAgent by requesting the public key of the MobileAgent from the KeyServer. The necessary time for mobile agent authentication, T_{MAAuth} , is given by:

$$\begin{aligned} T_{MAAuth} &= T_{reqdata} + T_{reqPK} \\ &+ T_{respPK} + (D_{state} * (1 / R_v)) \end{aligned} \quad (4.6)$$

Time for signing and encrypting the results to be provided by remote host, T_{Sdata} , is defined as

$$T_{SData} = D_{data} * ((1 / R_s) + (1 / R_e)) \quad (4.7)$$

The time for providing integrity and confidentiality is defined as:

$$\begin{aligned} T_{MASec} &= (D_{data} / R_s) + (D_{data} / R_e) + (D_{data} / R_d) + (D_{data} / R_v) \\ &+ (N * (D_{state} / R_v)) + (N - 1) * T_{SData} \end{aligned} \quad (4.8)$$

If the mobile agent visits N nodes, it needs to $N + 1$ migration and authentication. The total execution time for the secure SCAS, T_{SMA} , is defined as:

$$T_{SMA} = T_{MAth} + (N + 1) * T_{MAAuth} + T_{MASec} \quad (4.9)$$

The performance model can be used to achieve the desired performance by tuning the necessary parameters present in the execution environment.

5. CONCLUSIONS AND FUTURE WORK

This paper has attempted to improve the security aspects of mobile agents by proposing an approach based on closed network and agent tampering protection. Results show that the experimental, agent-based e-business system developed is sufficiently secure and that a performance overhead is introduced when the system is made to scale. Hence, a tradeoff between security and performance is identified. The performance model developed can be used to tune the real-time

execution parameters so that the desired level of performance with security can be achieved. The future work includes analyzing the performance of the system for different sizes of query and then simulating the behavior of malicious hosts.

REFERENCES

- [1] Butscgje, L.; Paprzycki, M.; and Ren, M. 2004. Mobile agent security – an overview, In: E. Niedzielska et al. (eds.), Modern Information Technologies in Management, Wrocław University of Economics Press, pp. 600-608.
- [2] Lange, Danny B., and Oshima, Mitsuru. 1999. Seven good reasons for mobile agents, Communications of the ACM, pp.88-89.
- [3] Hohl, Fritz. 1998. A model of attacks of malicious hosts against mobile agents, Proceedings of Fourth Workshop on Mobile Object Systems (MOS'98): Secure Internet Mobile Computations, <http://cuiwww.unige.ch/~ecoopws/ws98/papers/hohl.ps>.
- [4] Java Agent Development Environment. <http://jade.tilab.com/>
- [5] Zachary, John. 2003. Protecting mobile code in the wild, Internet computing, IEEE, 7(2).
- [6] Koliouisis, A., and Sventek, J.S. 2007. A trustworthy mobile agent infrastructure for network management. In, The Tenth IFIP/IEEE International Symposium on Integrated Network Management, pp. 383-390, Munich, Germany.
- [7] Hyungjick, Lee. 2004. The use of encrypted functions for mobile agent security, Proceedings of the 37th Hawaii International Conference on System Sciences – 2004, 0-7695-2056-1/04 \$17.00 (C) 2004 IEEE.
- [8] Massachusetts Institute of Technology. Kerberos: The Network Authentication Protocol. <http://web.mit.edu/kerberos/www/>
- [9] Lin, Min-Hui; Chang, Chin-Chen; and Chen, Yan-Ren. 2004. A fair and secure mobile agent environment based on blind signature and proxy host, Journal of Computer and Security 23(4), 199-212.
- [10] Karnik, Neeran M., and Tripathi, Anand R. 1999. Security in the Ajanta Mobile Agent System. Technical Report, Department of Computer Science, University of Minnesota.
- [11] Page, J.; Zaslavsky, A.; and Indrawan, M. 2004. A buddy model of security for mobile agent communities operating in pervasive scenarios, ACM International Conference Proceeding Series; Vol 54, Proceedings of the Second Workshop on Australian Information Security, Data Mining and Web Intelligence, and Software Internationalisation, pp.17-25.
- [12] Paprzycki, M., and Abraham, A. 2003. Agent systems today: methodological considerations, Proceedings of the 2003 International Conference on Management of e-Commerce and e-Government, Jangxi Science and Technology Press, Nanchang, China, pp. 416-421.
- [13] Rivest, R.; Shamir, A.; and Adleman, L. 1978. A method for obtaining digital signatures and public Key Cryptosystems, Communications of the ACM.
- [14] Sun Microsystems. Java Security Architecture. http://java.sun.com/products/jdk/1.2/docs/guide/security/spec/securityspecTOC_fm.html

- [15] Sander, Tomas, and Tschudin, Christian F. 1998. Protecting mobile agents against malicious hosts, *In*: Giovanni Vigna, ed., *Mobile Agents and Security*, LNCS 1419, pp. 44-60. Springer.
- [16] Tschudin, Christian F. 1999. Mobile Agent Security, *Intelligent Information Agents: Agent Based Information Discovery and Management in the Internet*, pp.431-446.
- [17] Wu, Xiaoping; Shen, Zhidong; and Zhang, Huanguo. 2006. The Mobile Agent Security Enhanced by Trusted Computing Technology, *Proceedings of International Conference*, pp.1-4.
- [18] Guan, Xudong; Yang, Yiling; and You, Yinyuan. 2000. POM-a mobile agent security model against malicious hosts, *Proceedings of High Performance Computing in the Asia-Pacific Region*, Vol. 2, pp.1165-1166.

ABOUT THE AUTHORS

A. Kannammal is a faculty member in the Department of Computer Technology and Applications at Coimbatore Institute of Technology, India. She received her Ph.D. in computing sciences from VIT University, Vellore, India, in 2007. Her research interests include electronic business, agent technology, and web services. She has published and presented papers in journals and at conferences. She serves as a reviewer for the *International Journal of Patterns, Software Architecture and Software Reuse*.

N. Ch. S. N. Iyengar is a professor in the School of Computing Sciences at VIT University, Vellore, India. He received his master of science degree in applied mathematics and his Ph.D. from the Regional Engineering College, Warangal (now known as NIT Warangal), Kakatiya University, Andhra Pradesh, India. He received his M.E degree in computer science and engineering from Anna University, Chennai, India. His research interests include fluid dynamics (porous media), information security models, e-business applications, agent technologies; QoS in networks; ontology-based web technologies and cryptography.