IJCSC International Journal of Computer Science and Communication Vol. 3, No. 1, January-June 2012, pp. 139-144

SECURED MOBILE AGENTS FOR MULTIPORT COMMUNICATION

Sreekanth V.[1], S. Ramchandram[2]

[1]Research Scholar, Jawaharlal Nehru Technological University Hyderabad, Hyderabad, India, *mail4sreekanth@gmail.com*

[2]Osmania University College of Engineering, Osmania University, Hyderabad, India, *s.chandram@gmail.com*

ABSTRACT

Conventional Mobile Agents has the capability to communicate with only one port or application on a host. These mobile agents are static and application specific. Mobile Agents in applications such as distributed computing, e-Auctions etc, an Agent that could communicate with more than one port/application is needed to come over limitations like reusing the agent reusability and optimizing network bandwidth. The objective of this paper is to demonstrate a Mobile Agent that is secure with reusable key, reusable agent and can communicate with more than one port/ application in a single visit to a host.

Keywords: Mobile Agents, Multiport Communication, Agent Security, JADE, RSA Key Encryption.

1. INTRODUCTION

The Mobile Agent is an autonomous software program that can move from one host to other in a network and perform a specified task. Its features like mobility, autonomy, size bring several advantages like less bandwidth requirement, limited latency and higher degree of robustness. Due to the mentioned advantages, Mobile Agents are not finding its applications in Distributed Computing, systems and applications along with electronic commerce and auctions.

Though the bandwidth requirement for an agent is less, further making the agent dynamic would reduce the bandwidth requirement so that the agent could hold more State information and data from multiple ports on a single host. The current generation agents are designed to communicate with only one port or application on each visit to a host. However, the agent can be made more dynamic if it can communicate with one or more applications in a single visit as need be instead of sending another agent to the same host.

Security is the other important aspect to be considered while communicating with remote hosts on a network. The data from the host is encrypted using a key; however, generating a new key each time would be costly and time consuming process. A reusable key is the solution to improve the performance of an agent.

In this paper, we discuss the key exchange technique of an agent and how an agent can communicate with multiple ports in a single visit to a host. The performance of these agents can be compared with the static agents currently being used.

Though the bandwidth requirement for an agent is less, further making the agent dynamic would reduce the bandwidth requirement so that the agent could hold more State information and data from multiple ports on a single host. The current generation agents are designed to communicate with only one port or application on each visit to a host. However, the agent can be made more dynamic if it can communicate with one or more applications in a single visit as need be instead of sending another agent to the same host.wide variety of systems require reliable personal recognition schemes to determine the identity of an individual requesting their services. Identity verification (authentication) in computer systems has been traditionally based on something that one has (key, magnetic or chip card) or one knows (PIN, password) [1]. Things like keys or cards, however, tend to get stolen and passwords are often forgotten or disclosed. To achieve more reliable verification or identification one should use something that really characterizes the given person. Biometrics offer automated methods of identity verification or identification on the principle of measurable physiological or behavioral characteristics like face, voice, fingerprints etc. Due to non intrusive and user friendly nature of the face biometric it is most commonly used in surveillance applications, crime investigations, security etc.

A fair amount of research work has been published on face authentication. Eigenfaces method proposed by Turk and Pentland in [2] uses a nearest neighbour classifier while feature-line-based methods explained by Li and Lu in [3], replace the point-to-point distance with the distance between a point and the feature line linking two stored sample points. In Fisherfaces method [4] authors uses linear/Fisher discriminant analysis (LDA /FLD). Bayesian methods proposed by Moghaddam and Pentland [5] use a probabilistic distance metric while SVM method uses a support vector machine as the classifier [6]. Utilizing higher order statistics, independent-component analysis (ICA) is argued to have more representative power than Principal

Component Analysis (PCA) on which [2] to [6] depends, and hence may provide better recognition performance than PCA [7]. Being able to offer potentially greater generalization through learning, neural networks/learning methods have also been applied to face recognition.

Earlier methods belong to the category of structural matching methods, using the width of the head, the distances between the eyes and from the eyes to the mouth, etc., or the distances and angles between eye corners, mouth extrema, nostrils, and chin top [8]. More recently, a mixture-distance based approach using manually extracted distances was reported in [9] by Cox et al.. Without finding the exact locations of facial features, Samaria [10] proposes a Hidden Markov Model based method which uses strips of pixels that cover the forehead, eye, nose, mouth, and chin. Nefian and Hayes [11] reported better performance than [10] by using the KL projection coefficients instead of the strips of raw pixels. One of the most successful systems in this category is the graph matching system [12], which is based on the Dynamic Link Architecture (DLA). Using an unsupervised learning method based on a self-organizing map (SOM), a system based on a convolutional neural network (CNN) has been developed by Lawrence et al. [13]. Pentland et al. [14] proposes a method that uses the hybrid features by combining eigenfaces and other Eigen modules such as eyes, mouth and nose are explored.

In the present work feature based face identification system has been discussed. The Gabor filter bank has been used to extract the features while neural network has been used as a classifier.

2. DEFINATIONS AND NOTATIONS

The integrity of an agent means that neither code nor the execution state should be changed by an unauthorized host, and if the code or execution state has been tampered with, such changes should be detectable (the owner, host or an agent platform, that wants to interact with the agent).

The authorized changes occur only when the agent has to migrate from one host to another. The formal definitions are given below:

Definition 1: (Integrity of an agent): An agent's integrity is not compromised if any unauthorized modification can be detected by the agent's owner.

Definition 2: (Strong forward integrity): None of the encapsulated messages m_k, with $k < n$, can be modified.

If a mobile agent visits a sequence of hosts H_1, H_2, …, H_n, and the first malicious host is after Hm, then none of the encapsulated messages O_k where $k > m$ can be modified.

Definition 3: (Publicly verifiable forward integrity): Any host H_i can verify that the chain of partial results $m(i_0)$, …. $m(i_{n-1})$ has not been compromised.

Table 1
Notations Used

O_0	Originator System of the Agent (Origin of Mobile Agent)
H_i (i > =1)	Set of hosts the Agent visits (intermediate Hosts)
m_i	Original Message made by host H_i
M_i	Encapsulated message from H_i
Hash(x)	Cryptographic hash function
r_i	Nonce generated by O_i
$MAC_{ki}(x)$	Message Authentication Code generated using secret key k_i

Each of these protocols has a common sequence of phases: Initial setup by originator, initial visit on intermediate hosts, and update of data and verification of data integrity. The agent visits a sequence of hosts H_1, H_2, ... H_n, and obtain the message m_i from each host, where an offer includes not only a price but some indication of the source of the offer, usually set of offers ω, a set of encapsulated offers Ω and key k, and we use subscript to denote these values over time. For example, ω_i represents the value of ω after the agent has visited the ith host.

3. INTEGRITY OF MOBILE AGENTS

To achieve strong forward integrity of data in the agent, an efficient key need to be generated and exchanged between originator and the hosts. Though we have different key exchange techniques, Deffie-Hellman key exchange technique is the most popular technique being used.

3.1 Deffie-Hellman Key Exchange

During the initial setup, each visited host $H_i > 0$ (i.e., each host except the originator) exchanges a secret shared key k_i with the originator O_o, using the Diffie-Hellman key exchange technique. The originator, O_o, then sends the agent to visit a set of hosts H_1, H_2, …, H_n with an initial set integrity value Γ_0 and empty data collection list ω_0.

Initial visit on a host:

MAC on the offer:

K_i = key from *DH* key exchange

$Hash_i = MAC_{Ki}(h_i)$

Limitations:

One of the limitations of this method is that a shared, secret key K_i is required between originator and each host that the agent visits. If the original key is compromised, the whole agent data is available to the malicious agent.

If the host is not known at the time of agent creation, then the originator must remain online to participate in this exchange method is vulnerable to 'man-in-the-middle' attacks, a fact that compromises the security of this technique.

3.2 RSA Key Exchange

RSA Key Exchange is a popular key exchange technique where the Originator O_0 generates an RSA public key OP_k and is sent to all the hosts in the network. Each host then generates its own key K_i, encrypts it with the public key of the originator and sends it back to the originator. Now that the Originator has the key the encrypted messages from each host is decrypted with the respective key of the host. The complete life cycle of an agent starting from the originator, collecting the message/bid from the host, performing partial encryption using the RSA key and moving back to the originator is shown in Fig. 1. The MAC is calculated on with the random nonce generated by the originator for each host r_i, actual message m_i, and the Host id H_i. This makes the partial encryption of the message from each host more secure and can be decrypted only by the Originator as only the originator knows both the key and the random nonce. The length of the key can be chosen depending on the complexity required by the application. As the key length increases, the time taken for encryption and decryption also increases.

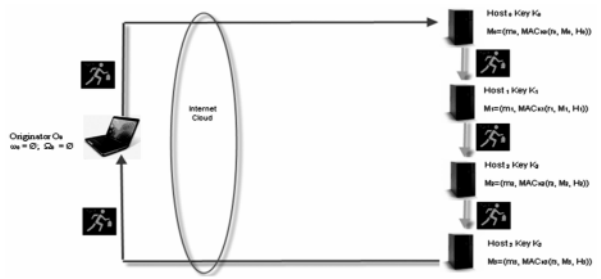


Figure 1: Agent Life Cycle, Key Generation & Encryption using RSA Key

Initial visit on a host:

MAC on the offer:

K_i = Host RSA key

$\text{Hash}_i = \text{MAC}_{HKi}(h_i)$

Advantages of RSA Keys:

The key can be reused for subsequent visits to the host, only initial key generation and message encryption is needed.

Though RSA encryption and decryption is costly, however, reusing the key in the subsequent visits makes it more affordable than Deffie-Hellman key exchange technique.

The size of the mobile agent is less using RSA keys as it only carries encrypted secret keys.

4. MULTIPORT COMMUNICATION

Conventional Mobile Agent is designed to communicate with only one application or port when it reaches the Host. However, making the agent more dynamic, an agent can communicate with more than one application or port. An Agent has 4 sections viz., Data, State, Code and Message.

Data corresponds to the Agents instance variable of a class in Object Oriented program. The data holds the information about the Originator, Host address; Host Keys, Nonce and the Port to communicate. Data section also holds the generic messages if any the Originator wants to share with the Hosts.

State is the object state directly controlled by the agent and holds the current execution state of an agent.

The actual logic of the agent is in the Code section. JADE is used as a Mobile Agent Platform for implementing Multiport Communication.

Message section holds the encapsulated partially encrypted message of each visited host.

4.1 Conventional Mobile Agents

A conventional Mobile is capable of communicating with only one port and holds a single encrypted message from each host. Figure 2 demonstrates the structure of the conventional Mobile Agent.

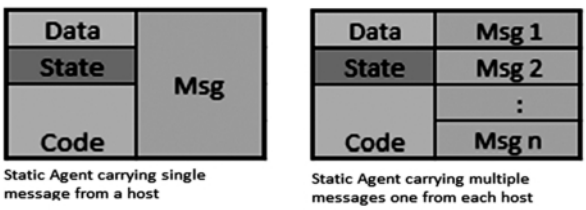


Figure 2

The conventional Agents have the following challenges,

Non-dynamic Agents: The agent can perform a single operation and carry the result or collect a single message from each visited host. While working with applications like distributed computing, an agent may have to communicate with multiple ports and/or work on a time sharing mode on the host.

Reusability: The agent is not reusable; a new agent is required for communication with an application or a port.

Efficiency: The efficiency of an agent is calculated by the number of messages sent in a given period of time, size of each message and the distance travelled. The efficiency of conventional agents has a huge scope of improvement.

The above mentioned challenges are addressed using Dynamic Mobile Agents.

4.2 Dynamic Mobile Agents

The data, state and code sections of an agent are modified such that an agent can communicate with more than one

port on reaching a host. The agent would be configured with the host address and all the ports it needs to communicate on a host. The execution state is repeated the number of times as the number of ports to communicate on each host. The agent is capable of working on a time sharing mode. Figure 3 shows the structure of the Dynamic agent.

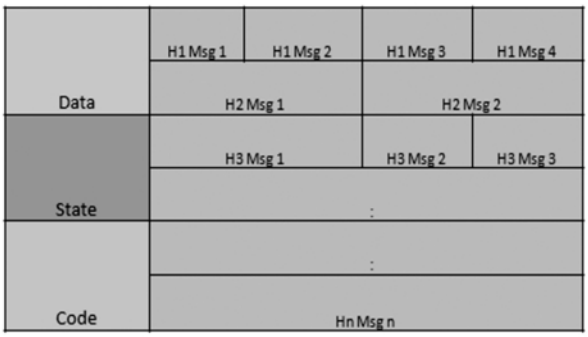


Figure 3: Structure of Dynamic Agent for Multiport Communication

4.3 Implementation of Dynamic Mobile Agents

The Originator is depicted as Main-Container and each host is depicted as a Container in the implementation phase to match the JADE terminology. The application is designed for the Agent to start from the Main-Container, visit Container 1 and 2 and collect the status of 2 different services from each host and return to Main-Container and update the status. Fig 4 below shows the status of Agents before Firing the Agent. The Status 'Visited Status' is marked "NO" indicating the Agent is in dormant state.

Each button shown on the system performs a specified action. The Fire button is used to start the agent from the main container and start visiting the hosts. Clear button would clear the status of the Service Status of the visited Containers (Hosts) and Refresh Locations button is used to set the status back to dormant state. Clone button can be used to Clone the Agent, but not used in the current demonstration as it is out of the scope of this article.

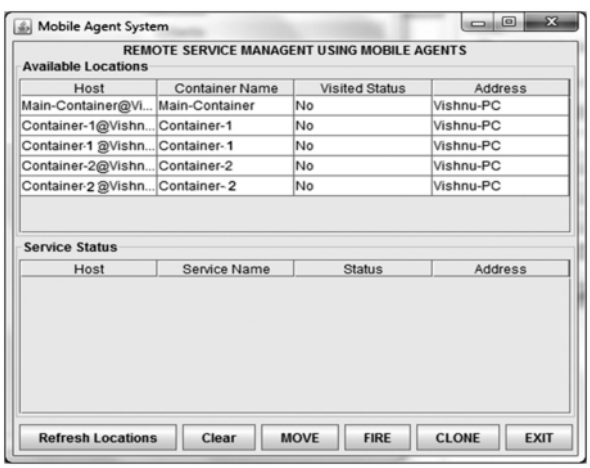


Figure 4: Container Status Before Fire

Figure 5 shows the status of each service after the Agent completes its iteration to Containers 1 & 2. The Agent check displays the status of services IP Helper and Netlogon from Container 1 and Fax & Print Spooler from Container 2. This application is an indication of an Agent performing more than one action on a Host.

Figure 5: Container Status After Fire

5. EXPERIMENTAL SETUP AND PERFORMANCE EVALUATION

5.1 Experimental Setup

The proposed system is implemented using Java Agent Development Framework (JADE). JADE simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that supports the debugging and deployment phases. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another one, as and when required.

The hardware used is 1 Intel Core i5 Laptops having 4 GB RAM and Windows 7 as the operating system. One window is set as an originator and the other windows are setup from 2 to 15 different Host Frames. The bidding system is designed in such a way that each host is visited at least once. The Key exchange server is setup on the originator system and designed to generate a 1024 bit key for each host.

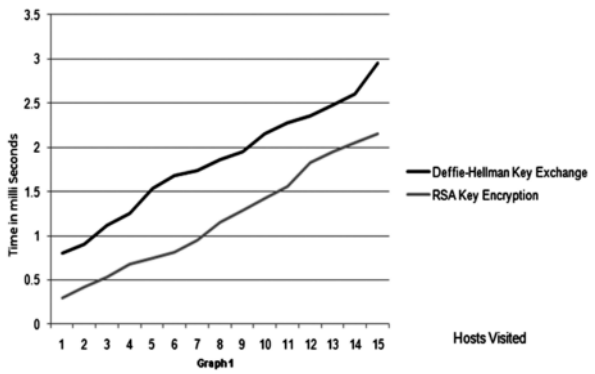
The implementation of this platform has been simplified since the platform already provides a Directory Facilitator (DF). The originator is setup to register for DF, the agent must include in its initialization (setup). The Host system sending bids is implemented using a Facilitator Agent (FA). When the agent receives a message, it adds the SearchBehaviour, that will do the search according to the type defined in the message and then it returns the results to the requester agent.

The application system is called Remote Service Management using Mobile Agents. The objective of this project is to control services of on a remote server using mobile agents. The functionality included obtaining the status of selected services. This application can be enhanced to Start, Stop and Restart services on the remote server from a host system using Mobile agent System.

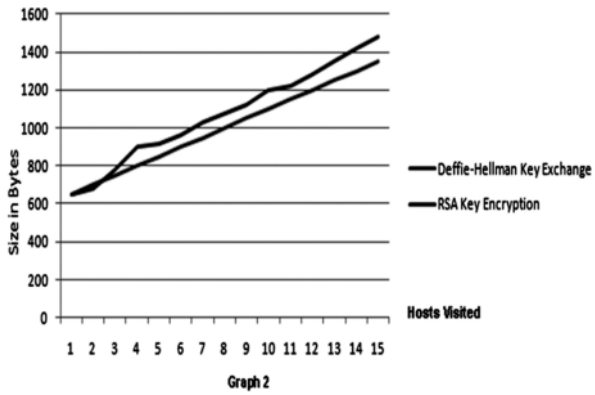
5.2 Performance Evaluation

The application is tested with the Agent visiting from 2 to 15 hosts and querying status of different services. The Agent is tested for querying 2 to 5 services on each host and the average latency of 50 ms at each host.

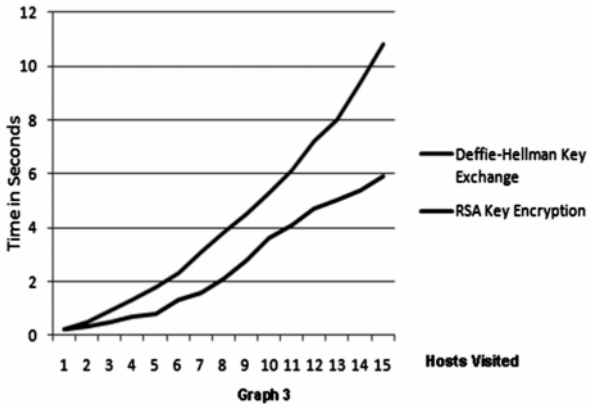
The first section of the performance evaluation is focused on the key generation time and the size of the agent during key exchange process. Graph 1 depicts that the key generation time between Deffie-Hellman Key exchange technique and the RSA Key technique. The graph clearly indicates the key generation time for RSA is lesser than the Deffie-Hellman key exchange technique. Though the RSA data encryption is costlier than the regular encryption technique, the key can be reusable.

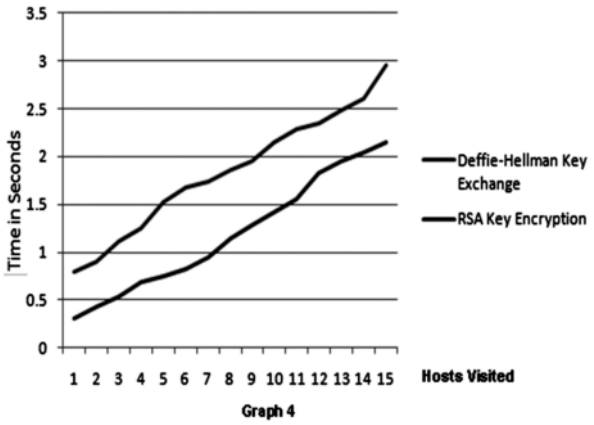


Graph 2 indicates the size of the Agent during key exchange. The key size can be determined by the programmer and in this setup; a 512 bit key is used. The size of the Agent while using RSA key exchange technique is comparatively less than Deffie-Hellman technique and thus decreases the cost and network bandwidth usage.



Graphs 3 and 4 depicts the performance of the Agents with a processing time of 500 ms and 50 ms respectively at each port. In either case, RSA Key encryption technique's perforce is getting better with the number of hosts visited when compared to the Deffie-Hellman technique.





6. CONCLUSION AND FUTURE SCOPE

This article discuss the security implementation of Mobile Agents using RSA Key exchange and getting status of services on a remote server/host using mobile agents. This application can be enhanced into a web based application that can control (Query, Start, Stop, Restart) services on a remote server. The agent architecture provided can communicate with selected multiple ports on one or more remote hosts or execute multiple commands on OS kernel and get the results using a single Agent. This application helps the Application Server administrators to get the status or control the services on remote server without logging on to the server.

The performance metrics indicates that the Dynamic Agent structure provided is utilizing less network resources and the faster than using the regular Static Agent. Reusing the keys and using RSA Key encryption makes the approach more secure and limits the size of the agent. The performance of this approach increases with as the number of host visited by the agent increases.

REFERENCES

[1] Jade, "Java Agent Development Framework". http://jade.cselt.it, 2004.

[2] V. Sreekanth, Prof. S. Ramchandram, Prof A. Govardhan, "A Novel Approach for Security and Integrity of Mobile Agents", *ICCBN 2008, IISc, Bangalore, India*.

[3] Alfonso Fuggetta, Gian Pietro Picco and Giovanni Vigna, "Understanding Code Mobility", *IEEE Transactions on Software Engineering*, **24(5)**, 1998.

[4] W. Diffie and M.E. Hellman. "New Directions in Cryptography". *IEEE Transactions on Information Theory*.

[5] G.karjoth, N.Asokan, and C. Gulcu. "Protecting the Computation Results of Free-roaming Agents". *In 2nd International Workshop on Mobile Agents*, **1477** of Lecture Notes in Computer Science, pp. 195-207. Springer-Verlag, 1998.

[6] Roth. "Empowering Mobile Software Agents". *In Proc. 6th IEEE Mobile Agents Conference*, **2535** of Lecture Notes in Computer Science, pp. 47-63. Spinger.

[7] J. Buchmann. "Introduction to Cryptography 2nd Edition". *New York Springer-Verlag*, 2004 pp. 289-292.

[8] S. Loureiro. "Mobile Code Protection". *PhD Thesis, ENST Paris/Institute Eurecom*, 2001.