A Consistent Protected Structural Design for Mobile Agents in Open Network Systems

Arihant Khicha, Neeti Kapoor

Abstract- A system in which user programs (the agent) may willingly and separately travel from one the host to the mobile agent server is a mobile-agent system. A large exploitation of mobile agent systems is not possible without gratifying security structural design. The attack of a visiting code by a malicious host is the major barrier facing wide exploitation of mobile agents. The fact that host computers have complete control over all the programs of a visiting agent makes it very hard to protect agents from untrusted hosts. This has resulted to restricted exploitation of mobile agents to acknowledged hosts in congested networks where the agent's security is assured. However, this restriction negates the original major concept of sovereignty on the basis of which mobile agent technology is established. This paper proposed a dynamic protected structural design for mobile agents systems, using Platform Registry and Travel Diary Protection Scheme. The scheme protects and allows mobile agents to travel liberally in open networks environment.

Keywords- Mobile Agents, Travel Diary, Security, Platform Registry.

I. INTRODUCTION

Mobile agent is an agent that can be simply consider as a unit run in dynamic environment with independent ability and mobility. This technology has great significance in data mining, e- Commerce, distributed computing, network management etc. Security is the main issue that prevents mobile agent from being widely used. There are two aspects in which Security problems lie: security of host and security of agent.

The first difficulty has many common characteristics of conventional computer security; so to solve the difficulty with satisfactory results equivalent conventional methods are used. But the second difficulty is still a major challenge. Our study concentrates on the mobile agent's security on a host platform. Moreover, an agent can get on two categories of expedition: the first is expedition with a diary containing pre-defined itinerary and second is expedition in which the mobile agent has no fore knowledge of the host to visit. It is called a free-roaming mobile agent whose protection is more difficult. Methods used to defend an agent's data and its state count on the category of the agents' expedition.

These free-roaming agents can face more complex attacks such as replay attacks, colluded truncation attack, or many other host attacks on a visiting agent if traveling dairy does not specify where to visit.

This paper focuses on security of free-roaming agents in open net environments and it also presents a security protocol which performed a good role in preventing attacks.

II. LITERATURE ANALYSIS

Many security issues have been identified since beginning of mobile agent. These issues were classified according to the unit being attacked: attack of agents against agents, attack of agents against hosts, and attack of hosts against agents and the source of the attack. The first type is categorized as – attack of **agents against agents** in which we find attacks where the agents access or modify another agent's data, masquerade their identity in order to make a transaction forged, or repetitively it also send messages to another agent in order to initiate a denial of service attack, among others. The second type is attack of **agents against host** and it includes threats in which agents use system resources unnecessary, access resource which they can access as well as perform malicious action, expand access to a service to which they are not permitted, and so on. For the first two categories, where agent is an attacker already a sound solutions is proposed. Along with the solutions that provide a satisfactory level of protection, the most resourceful is Software-Based Fault Isolation [3]. This method, also known as sandboxing based on limiting program accessibility in a congested field, in such a way that the available resources and program address space are restrained within this field. The different method for these kinds of attacks consists: using safe code interpretation [4], where it prevents the agent from attacking the host by using the set of available instructions: providing the authentication to the agent owner by signing the code, as well as also include some method to check the owner level of trust [10], , in order to proof that the implementation of that code is secure [11], along with the code it also send logical demonstrations. Concerning the second category - agents **against host, in this** any peripheral entity can be source of the attack if it is also not part of the agent platform. This peripheral entity can perform attacks against the host's communications with the outside or against the platform resources (files, communication ports, etc.). In these cases, to a great extent, the mechanism on which security depends is provided by the operating system.

Transport Layer Security [12], secure communication channel, which is established using mechanisms is used to provide the secure communication between the host and other parties. The third category that is **host against agents** is very difficult to prevent. It is evident that if a host wants to execute an agent, it must have complete access to the agent state, code, and data. It is difficult to prevent the host to analyze the agent code, corrupting its state or data, modifying its execution environment, or execute it multiple times, for example, generate multiple purchases in a shopping scenario.

Manuscript Received October 18, 2011.

Arihant Khicha,Information Technology,RIET,Jaipur, India. 9001099903, (e-mail: arihantkhicha@gmail.com).

Neeti Kapoor, Information Technology, RTU, Jaipur, India. (e-mail: neeti.kapoor87@gmail.com).

If we want to keep an agent's data secret to prevent it from the host, then it must be stored in a way that even the agent itself cannot directly access it encrypted with the key of a different host platform.

To deal with the malicious host server problem several mechanisms have been projected. Few of the solutions of the malicious host problem are impractical. They have been designed for particular situation that are actually rarely found in real-life applications. Some of the better known ones are:

- Execution tracing
- Obfuscation
- Computing with encrypted functions
- Tamper-proof devices

A. Execution Tracing

Execution tracing [13] is a procedure that allows unauthorized modifications of an agent to be detected upon completion of the agent execution. The protocol proposed in [14] records the agent's behavior on each platform to trace its execution. The trace consists of a sequence of identifiers according to the operations executed by the agent. Platforms maintain and produce traces of all executed agents, and after the termination of agent's execution agent owners can request these traces and from this it can be verified that the agent code or state is maliciously modified or not. But this approach has several drawbacks, such as the number of logs and size to be reserved by platforms, or once the agent has returned to the home platform there is a lack of connection between the owner and the platforms. Moreover, the verification mechanism is only used when the owner has a disbelief that the agent execution has been corrupted and this mechanism is too expensive to be applied systematically.

B. Obfuscation

The aim of Code obfuscation [13] is to generate executable agents who cannot be attacked by manipulating or reading their code. This procedure transforms the agent code in such a way that it is functionally identical to the original one. There is also a time interval during which the agent and its sensitive data are applicable. After this time elapses, any attempt to attack the agent becomes useless. The main limitation of these techniques is that an attacker gets difficulty in establishing the time to understand an obfuscated code. Likewise, there is no mechanism currently known by which an agent quantify time to accomplish its task, especially in heterogeneous environments. As a result, restricting the lifetime of a mobile agent is not feasible in practice.

C. Computing with Encrypted Functions

It is a technique proposed by Sander and Tschudin [12] to achieve code integrity and code privacy. In this technique encrypted programs are created that can be executed without decrypting them. If a mobile agent execute a certain function f then that function f is encrypted to obtain E (f) and a program is created that implements E (f). Platforms execute E (f) on a clear text input value x, without knowing what function they actually computed. The execution yields E (f(x)), and only the agent owner can decrypted this value to obtain the desired result f(x). The main limitation of this technique is that the encryption schemes are applicable for polynomials, using function composition techniques and homomorphism encryption. Thus, their proposal is not suitable for general programming.

D. Tamper-Proof Devices

A tamper-proof device is based on the entire agent execution on a physically sealed environment, which can be trusted to execute the agent correctly. Tamper-proof devices are provided by a trusted third party and they can be checked from time to time to verify that their security has not been compromised. It can be used to perform cryptographic operations with a private key that must be kept secret from the remote host. They can also have their own private key, for example, to sign partial results generated by the agent. This approach has two limitations: the costing of tamper-proof device on every platform. Secondly, the approach is only suitable for closed environments, such as corporate networks such as within in a group of banks in a geographic political area. As a result, the technique implies a loss of agent sovereignty. Hence this paper focuses
on pragmatic protocol that solves the malicious host problem.

III. OVERVIEW OF MOBILE AGENT ARCHITECTURE

Figure 1 to 5 given below specifies the structure of mobile agent systems. In a network environment Mobile agents move around it and visit every computer, hopping from one host to others. Mobile agent servers handle the execution of the program code and dispatch agents to different computers. each agent has its own thread and host severs execute it. Through message any communication is done between agents of different servers.

A. Mobile Agents Interactions an a Server

Figure 1 specifies the relationships between various agents and also show that how they complete the execution tasks through communicating using messages. The resources needed by visiting agents is provided by host

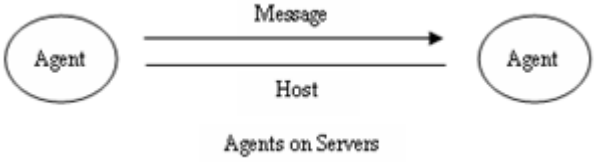

Fig. 1

B. Mobile Agent Server Architecture

The structure of a mobile agent server is shown in Figure 2. The activities of the visiting quest agents are coordinated by the server.

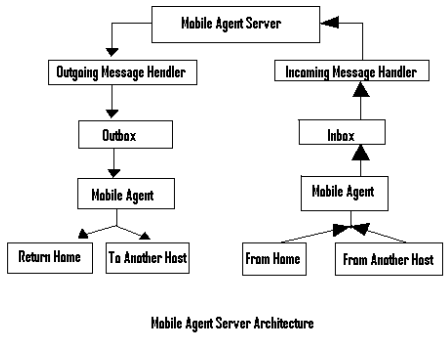

Fig 2

C. Transfer of Mobile Agents between Servers

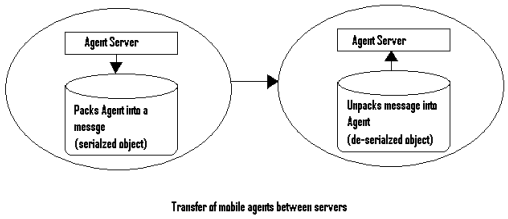

Fig 3

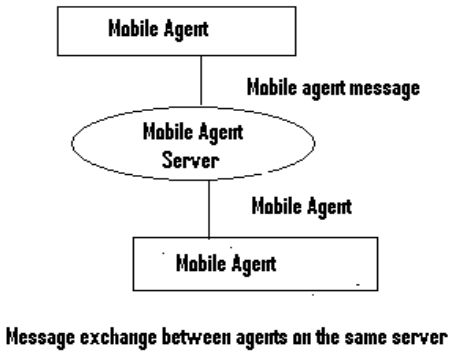

Fig. 4

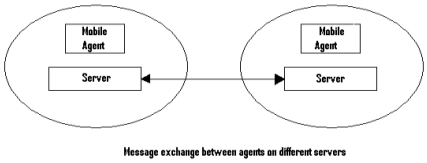

Fig. 5

When an agent completes its task on a host, it either switch to another host platform or returns to its home host. The figure 3 given below shows how object serialization/de-serialization is used in dispatching agents from one host to another .The process of converting a data structure or object into a format that can be stored (for example, in a file or memory buffer, or transmitted across a network connection link) and "resurrected" later in the same or another computer environment [14] is called Object serialization. In the mobile agent communication Messages passing form an important component; semantically identical clone of the original object is created by message passing. De-serialization is the process of restoring the object.

D. Message Exchange between Agents on the Same Server

The exchange of messages between two or more agents on the host and between different hosts is facilitated by the host platform .The roles of the server in intra/inter server agents' communication is shown in Figure 4 shows while figure 5 illustrates message exchange between agents on different servers.

E. Agent's Security Challenge

The mobile agent architecture given in figures 1 to 5 shows that agent has to face many security challenges while moving through the network to perform its duties. A lot of research has been done to solve the security problems in mobile agent systems. This research differs in its aim, emphasis, base, and technique. Some works concentrates on building the foundations for the security of a mobile agent system; some offer security mechanisms following different approaches; some work introduce security mechanisms into the architectures of mobile code systems; and others implement real applications with security features.
Nevertheless, these research works didn't provide any protection framework for protecting mobile agents on the host server they execute on. This is the problem that this work addressed.

IV. THE APPROACH

A. Agent's Itinerary

There are several protocols which protect theagent's itinerary. In these protocols the itinerary information is stored in a separate data structure, and after that cryptographic mechanism is used to protect this data structure. If this information is maintained and stored outside the main agent code, it is said to be explicit, and its protection is considerably simplified.
However these protocols do not support the protection of free-roaming agents. Agents then enforced to travel static itineraries, in which all host are known in advance. On the other hand, for free roaming agents the most functional and realistic mobile agent-based applications should be based on using dynamic itineraries, where some host platforms are discovered at runtime.

B. Securing Dynamic Itineraries

To support free-roaming agents, we use a protection scheme in which trusted locations are introduced into the agent's route. The information associated with dynamically located host can be stored by introducing some trusted hosts into the itinerary in our architecture.

Platform Registries

Platform registries are digital security infrastructures, maintained by trusted certificate authorities, such as Entrust Secure Server Certification Authority, RSA Data Security Inc, Baltimore Cyber Trust, , Equifax Secured Certificate Authority, , e.t.c. It is use for the registration and insurance of trusted digital certificates to public mobile agents' host platforms

Assumptions Made with Regard to Trusted Platforms

The main purpose to introduce trusted platform registries into agent's itinerary was to execute the agent task on the expected platform. The structural design presented in this scheme assumes that a agent's task is executed by trusted platform honestly. Furthermore, it is assumed that to prevent attacks from third parties who alter the agent execution the trusted agent platforms are protected with appropriate mechanisms. In this scenario, security depends on the mechanisms provided by good design of associated protocols and the operating system .These protocol also assumes the existence of a security infrastructure that allows users and agent developers to conclude whether a platform is reliable or not. An example of this security infrastructure can be found in [14]. In this work, the authors illustrate a security structure for a mobile agent system which incorporates a simple trust model. Such model establish trust relationships in a way similar to that used to handle distributed authentication in public key infrastructures

The identification of reliable platforms can also be ashore on simpler mechanisms, such as relying on real world trust relationships. For example, the platform from which the agent was first launched or the platform associated with a bank where the user has an account, can be safely introduced into the agent's itinerary as trusted platforms.

C. The Protection Architecture

This protection architecture intended to protect flexible dynamic mobile agent itineraries. There are three main objectives of architecture:

- Integrity: Platforms is not able to modify the agent's itinerary gradually.
- Authenticity: Platforms is not able to verify the identity of the agent owner.
- Confidentiality: Platforms is not able to access itinerary information of other platforms.

The Idea

The general initiative of this protection architecture is to construct a chain of digital envelopes, which contain two elements: the data, and the encrypted key which allows the decryption the following envelope. The proposal is illustrated in figure 6 below.

The entry of the protected itinerary is shown in this figure 6 envelopes. Each envelope, (ej) is encrypted using a random symmetric key (kj), and further this symmetric key is encrypted using the public key, (pj) of the host j, permitted to open the envelope. Thus, only the intended host can decrypt each envelope. Moreover, the symmetric key used to decrypt an envelope is protected inside the previous envelope so the envelope is only opened in the correct order.

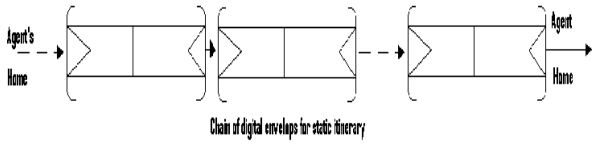

Fig. 6

D. Supports for Dynamic Itinerary

It is impossible to build a chain of digital envelopes if in the problem of protecting dynamic itineraries all public keys are not known in advance. In particular, the agent discovered the hosts that will be visited to execute an itinerary dynamically at runtime. Therefore, when the itinerary is created, the public keys of such platforms are not obtainable. So to solve this problem, a novel protection scheme is developed based on the agent itinerary protection, on the dynamically discovered platforms, using the public keys obtained from their corresponding platform registries. This proposal use platform registries discussed above by changing the chain of digital envelopes, as shown in figure 7.

When an agent asks a dynamically located platform for the purpose of obtaining its public key, for its platform registry identifier, and this host fallaciously gives a wrong id, then it is either the host itself will be unable to decrypt the message meant for it or the registry will be unable to supply its id. In either of the two cases, the agent is protected from this malicious host.

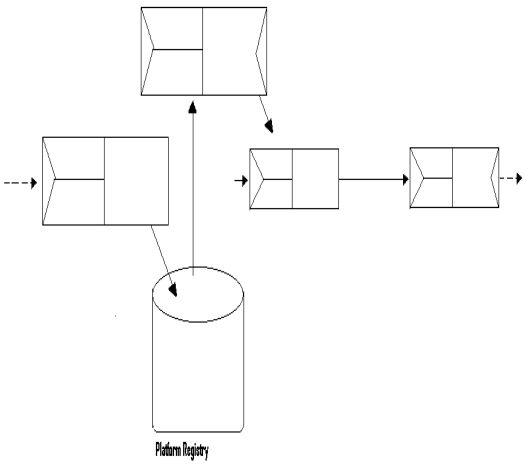

Fig. 7

E. Securing Itinerary

For every itinerary host, a random symmetric key *k*1, *k2*, *k*n is created to secure an itinerary. From a host i to j, each possible migration is denoted by t_{ij} is constructed as

$$t_{ij} = a_j P_i (s_i (id\ k_j\ a_j)) \qquad (1)$$

Where p_i refer to an asymmetric encryption function using the public key of platform i, a_j refer to the address of host j , s_i refer to a digital signature function using the private key of host i

From the equation, the transition from host i to host j contain the random symmetric key K_j associated with host j. When going to host j this key is used to encrypt the envelope of the protected itinerary. Platform j is encrypted using the public key of host j to ensure that it has access to K_j . Finally, t_{ij} consist of a unique agent identifier *id* that is used to thwart from replay attacks. Both *id* and K_j are signed by the agent's owner so that host j will be able to verify the agent's identity and integrity of the information it carries. The equation 1which is used to build agent transitions t_{ij} from host i to j is useful only when the host j is not dynamically located at runtime, that is a_j and the public key of host j is known at the agent home before the start of migration. When host j is located at runtime, then

$$t\ i\ j = a_j? \qquad (2)$$

Therefore $P_i(s_i(\text{id } k_j\ a_j))$ is replaced with ?, an unknown value. This is because at the time of creating the itinerary a_j is not known and the public key needed to compute $(s_i(\text{id } k_j\ a_j))$ is not available. In this scenario, when the host is located at runtime, its

address a_j and its corresponding agent Platform Registry Identifier ($regId_j$) will be obtained from this host for the reason of getting its equivalent public key needed to calculate k_j

The transition from the current host i to the dynamically located host j is now calculated as

$$P_j = platformReg(a_j\ regId)$$
$$t\ i\ j = a_j\ P_i(s_i(\text{id } k_j\ a_j)) \qquad (3)$$

The equation 3 is correspondent to equation 1 above. The main difference is that k_j is replaced with the random symmetric key for the new host, a_j , generated from the public platform registry access function that takes an agent host address and its corresponding platform registry identifier and return the host public key, if the host is registered with the registry and null otherwise. To obtain a host's public key, p_j from the host directly is not safe.

The agent owner digitally signed the symmetric keys *k*1, *k2*. *k*n , which are used to encrypt the entries of the protected itinerary. It ensures that attackers can neither modify existing ones nor generate their own itinerary entries.

Also, the entries previously generated by the same owner are prevented for reuse by the unique agent identifier id. Therefore the integrity of the protected itinerary is guaranteed. Moreover, every transition to a host j includes address a_j of the host. Consequently the hosts can verify that they were really part of the itinerary.

F. Simulation

We performed and implement two multi-phased experiments to prove the viability of the proposed structural design. Experiments first part was based on the proposed security architecture, simulating a simple mobile agent-based application on a hotel search and reservation system, using a local area network (LAN) of thirteen computers, with three serving as platform registries and ten serving as host servers. Each of the ten computers was setup to act as mobile agent server to their respective hotels and configured with appropriate programs to make them malicious and very hostile to visiting mobile agents. The user preferences are considered with regard to room facilities and guest services and according to that the system allows an individual to find the cheapest hotel in a given destination. The application allows the user to define search criteria. After defining the search criteria, to obtain a list of the five cheapest hotels in the destination, a mobile agent is started querying a remote hotel search engine.

The agent then visits each one of these hotels and then the room availability is checked for the desired rates, as well as their room facilities; services, etc are also checked. Besides this, a special discount is also negotiated by the agent for long stays. Our dispatched agent randomly visited eight of the ten servers and after execution log on each server visit it eventually returned to home.

Experiment second part was the same to the first except that the dispatched agent employed obfuscation methods for its itinerary without the proposed new protection scheme.

Table 1

Number of Mobile Agents on Hotel Reservation Assignment	Mobile-agent without Platform Registry Protection Protocols	Mobile agent with Platform Registry Protection Protocol
13	Altered Unaltered Killed	Altered Unaltered Killed

G. Analysis of the Log Files

In our experiments the agent code was designed to return its execution log on each server visited. By this we can analyze its venerability to attacks by its hosts. Using this same proposed protocol the execution log files at each server were encrypted with the public key of the agent home platform. By analyzing the log files showed it is clear that no successful attempts were prepared to read the agent's itinerary which included the packets for the next host to be visited from the current server and packets from the agent's previously visited servers. The packet that was previously encrypted with its own public key obtained from one of the platform registries could only open by the current host server. While, in our second experiment, where we don't used obfuscation methods with our proposed protection architecture, the analysis of the returned log files show that four of the host servers visited was able to access the agent's data packets that were not meant for these hosts. Table 1 show mobile agents with and without platform registry protection scheme. As shown in table 1, our proposal made it difficult for the host to alter the packet.

In relation to time performance factor, the execution time of the agents with our proposed scheme was compared to the execution time of the agents in our second experiment, that is, the roaming agents without our proposed protection framework, to determine if the proposed protection architecture increased the execution times considerably. We find that approximately 40.6%, the execution time of the agents with our protocol is increased in comparison of the unprotected agent's execution time as shown in figure 8.

This increase is largely due to the time required to execute complex cryptographic protection protocol at platform registries and on each of the host platforms visited. We also found out that the time increase is a linear function of the number of hosts visited. If the actual task to be performed by an agent on each server is itself complex and time consuming then the increase in time would be negligible.

V. CONCLUSION

In open network environment this paper proposes the use of a chain of digital envelopes with platform registries to support dynamic agents' itineraries. This proposal prevents a host server to gain access to the information carried by a mobile agent that is not meant for it, that is, the current host. In terms of data integrity and security ,this proposal display better performance when compared to the results obtained from obfuscation methods. However, in comparison to obfuscation methods this offer consume a little more time in visiting platform registries and to execute complex cryptographic functions .

REFERENCES

1. Wahbe R., S, Lucco, T.E. Anderson and Graham S.L., 1993, Efficient Software Based Fault Isolation, In Proceedings of the 14th ACM Symposium on Operating Systems Principle, pp 203-216, ACM
2. Jacob Y. Levy, John K. Ousterbhout and Brent B Welch, 1997, The safe Tcl Security Model Technical Report, Sun Microsystems
3. Sreekanth V., S Ramchandram and A. Govardhan, 2010, Mobile Agent Security and Key Management Technique, Journal of Computing, Vol. 2, Issue 9, ISSN 2151-9617
4. Neelesh Kumar Panthi and Chaudhari Neelesh Kumar Panthi, 2010, Securing Mobile Agent using Dummy and Monitoring Mobile Agent, International Journal of Computer Science and Information Technologies, Vol 1 (4), pp 208-211
5. Sarvarnl Islam Rizvi, Zinat Sultana, Bio Sun and Mid Washiqul Islam, 2010, Security of Mobile Agent in Ad Hoc Network using Threshold Cryptography, World Academy of Science, Engineering and Technology, Vol 30, pp 424-427
6. Sreekanth V., Ranchandra S., and Gavardhan A.,2008, A Novel Approach for Securing and Integrity of Mobile Agents, ICCBN, IISC, Bangalore
7. Tomas Sander and Christian F. Tschudin, 1998, Protecting Mobile Agent against Malicious Hosts, In Giovanni Vigna, Mobile Agent Security, pp. 44-60, Springer-Verlag, Herdeberg Germany
8. Gray R.S., 1995, A Transportable Agent System, In proceedings of CIKM 95 Workshop on Intelligent Information Agents 14
9. Dierks T. and Rescorla E., 2006, The Transport Layer Security Protocol Version, In RFC 4344, IETF
10. Vigna G., 1998, Cryptographic Traces for Mobile Agents, In Mobile Agent and Security, Vol., 1419 of Lecture Notes in Computer Science, pp 137-153, Springer Verlag
11. Hohl F., 1998, Time Limited Blackbox Security: Protecting Mobile Agent from Malicious Hosts, In Mobile Agent and Security, Vol., 1419 of Lecture Notes in Computer Science, pp 92-113, Springer Verlag
12. Sander T. and Tschudin C.F., 1998, Protecting Mobile Agents against Malicious Host, In Mobile Agent and Security, Vol., 1419 of Lecture Notes in Computer Science, Springer Verlag
13. Tan H.K. and Morean L 2001, Trust Relationships in a Mobile Agent System, In Mobile Agent, Vol., 2240 of Lecture Notes in Computer Science, pp 15-30, Springer Verlag
14. Carles Garrigne Ollivera Bellaterra, 2008, Contribution to Mobile Agent Protection, PhD Thesis, Universtat Ant Onoma, De Barcelona
15. Wikipedia the Free Encyclopedia Serialization, http://en.wikipedia.org/wiki/serialization

AUTHORS PROFILE

Mr. Arihant Khicha, M.Tech , Mca from MSRIT , Bangalore, Pursuing his Ph.d , written 5 books on Database management system, Operating system and Advance Database management system. He has also attended various seminar on Networking.

Neeti Kapoor, Btech done her Engineering from RCEW Engineering college Jaipur and Pursuing her M.Tech from Arya college of Engineering and Technology. She has written various books on Database Management system , operating system .