A Security Model for Data Storing and Data Collecting Agents

D.S.Adane[1] and S.R.Sathe[2]

Department of Information Technology
Shri Ramdeobaba Kamla Nehru Enginnering college, Nagpur, India

Department of Computer Science Engineering
Visvesvaraya National Institute of Technology , Nagpur, India

Abstract

Communicating with confidential data requires special attention in a Mobile Agents environment, especially when the other hosts must be prevented from eavesdropping the communication. We propose a communication model for secured communication between the agents belonging to Publishers and Consumers. Data confidentiality is ensured using our on the fly Encryption-Decryption sequence using ElGamal system to directly convert the message or plaintext into one that is encrypted directly with the public key of Consumer. The scheme ensures that the data possessed by the agents is secured at all times when it is executing at any of the untrusted hosts. We also explain how the homomorphic property of ElGamal scheme can be integrated with our model for a web based application like voting involving multiple agents.

Keywords: Mobile Agents, ElGamal, Confidentiality.

1. Introduction

As the agent technology evolves, awareness of security for this kind of technology is increasing. Providing security to Agents code and the sensitive data it carries is still a challenging task. The notion of an agent roaming around the network carrying its code, state and data and executing on foreign (often untrusted) host, makes it an easy target for security violation as it completely at the mercy of the foreign host on which it sits and execute. Thus, main problem in providing security to Mobile Agents is the fact that the execution environment does not belong to the user who has created the agent. As such after the agent is launched, the user does not have any control over the agent. This aspect also makes it difficult to apply the traditional security measures to this computing paradigm. A security solution is therefore required which would guarantee the security of agents code and its personal data. Over the years research has been done in the area of providing security to mobile agent platform and the agents themselves [1]. Interesting security measures, as mentioned in [1] and [2], for agent platform include Software-Based Fault Isolation, Safe Code Interpretation, Signed Code, State Appraisal, Path Histories and Proof Carrying Code. Similarly, various security mechanisms for protecting agent themselves include Partial Result Encapsulation, Mutual Itinerary Recording, Execution Tracing, Environmental key Generation, Computation with encrypted functions and Obfuscated Code. As, per survey [1], "the area of mobile agent security is still in a somewhat immature state. The traditional host orientation toward security persists, and the focus of protection mechanisms within the mobile agent paradigm remains on protecting the agent platform. However, emphasis is moving toward developing techniques that are directed towards protecting the agent, a much more difficult problem. Fortunately, there are a number of applications for agents where conventional and recently introduced security techniques should prove adequate, until further progress can be made."

With the above observation in mind in this paper we are dealing with providing security to agents personal (confidential) data using standard ElGamal encryption [3] and we also propose a communication model for the proposed security scheme. We feel that this model would be useful for many real world applications such as publishing and collecting the data on the web, voting or polling. Next section describes the assumption of our mobile agent model. This is followed by brief description of our encryption scheme [4]. Lastly, we give the details of our model and the integration of our encryption scheme with it. We then elaborate on how the homomorphic property of ElGamal system can be utilized with our model for specific web based application like voting.

2. Mobile Agent Model

A number of models exist for describing agent systems [5, 6, 7], however, for discussing security issues it is sufficient to use a very simple one, consisting of only two main components: the agent and the agent platform. An agent comprises the code, state and data needed to carry out some computation. Multiple agents cooperate with one another to carry out some application. Mobility allows an agent to move or hop among agent platforms. The agent platform provides the computational environment in which an agent operates. The platform where an agent originates is referred to as the home

Manuscript received April 5, 2009
Manuscript revised April 20, 2009

platform, and normally is the most trusted environment for an agent. It is assumed that the platform can eavesdrop on the agents data and communication hence confidentiality is required for both. It is also assumed that the platforms would not collude to compromise the data. An agent platform may support multiple locations or meeting places where agents can interact. Figure 1, which depicts the movement of an agent among several agent platforms.

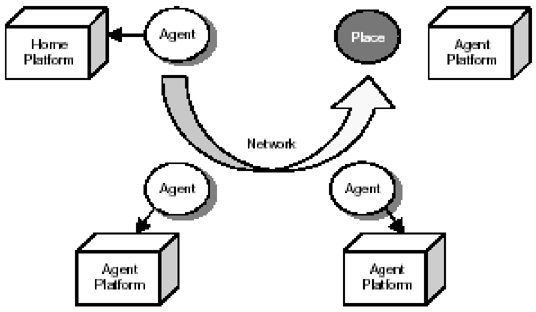


Fig. 1 Agent System Model

3. Encryption Scheme Used

In this section we briefly describe our encryption scheme [4] based on ElGamal system, to be used in the communication model. The typical problems in communicating confidential data through agents are:

a) It is not known beforehand who the agent will communicate with, thus the data cannot be encrypted with the proper key of communicating partner.

b) The environment in which the agent is working may be untrustworthy. Thus the data agent carries must be kept confidential all the time.

In conventional Client-Server systems the data is usually encrypted with the data owner's key to keep it confidential and when the data is needed in communication it is .first decrypted and then again encrypted using the public key of communication partner or the session key used during the communication. In an agent environment this is not an acceptable solution as the data is at one moment unencrypted and accessible by the host (untrusted host on which the agent resides). In our scheme, the data is first encrypted using the encryption key of the agent. At the moment data must be exchanged to another party, the data is again encrypted, but this time with the encryption key of the communicating partner. A decryption process then follows where the decryption key of the agent is used, such that the overall result is encrypted data, which can only be deciphered by the communicating party. This solution is referred as **E-E-D**. The process is depicted in figure 2 below:

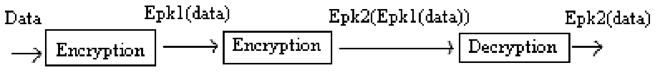


Fig. 2 Confidentiality in agent communication.

A necessary condition for an encryption algorithm to be used as E-E-D is:

$$DSK1\ (EPK2\ (EPK1\ (M))) = EPK2\ (M) \qquad (1)$$

where, PK1 and PK2 are the public keys of the agent and communicating party respectively. SK1 and SK2 are their corresponding private keys. It is assumed that there are more than one secret keys generated by the agent corresponding to different types of data. Initially the data to be encrypted is stored at the users computer and in order to encrypt it, the user .first generates a key pair for the agent according to the ElGamal system depending on type of data. The user generates a large random prime p and a generator α of the multiplicative group Z^*_p of the integers modulo p.

The user selects a random integer a1, 1<=a1<= p - 2, and computes :

$$y1 = \alpha^{a1} \bmod p \qquad (2)$$

The agent's public key is (p, α, y1) and its private key is a1.

The user encrypts the data (represented by parameter m) as follows. He first selects a random integer k1, 1<= k1 <= p – 2 and computes:

$$\gamma 1 = \alpha^{k1} \bmod p;\ \delta 1 = my_1^{k1} \bmod p \qquad (3)$$

The cipher text is c1 = (γ1, δ1). This is stored in the agent and can be run on any platform at any host. At the moment that the agent needs to give the personal data to another entity in the system, the following process is started. The agent collects the communicating partner's (from here on called Bob) public key y2, which is formed in the same way as the user's public key ($y2 = \alpha^{a2}$ mod n). Bob's private key is a2. It must be noted here that in order to fulfill equation (1), Bob must use the same generator and prime number for generating its key pair as the user. The agent encrypts the cipher text c1 using Bob's public key y2, by the following computations:

$$\gamma 2 = \alpha^{k2} \bmod p;\ \delta 2 = \delta 1 y_2^{k2} \bmod p \qquad (4)$$

Where, k2, 1 <= k2 <= p - 2, is an integer chosen at random by Bob. The second cipher text c2 is then formed by the pair (δ2, γ1).

It is now possible to decrypt it once using the agent's private key:

$$m' = (\gamma 1^{-a1})\ \delta 2 \bmod p = m y_2^{k2} \bmod p \qquad (5)$$

The result is an encryption of m based on PK2, e.g. y2. This is sent to Bob, who can decrypt according to the normal AlGamal decryption:

$$m = (\gamma_2^{-a2})\ m' \bmod p \qquad (6)$$

Decryption of m' (6) should occur at different place than the one where E-E-D operation took place as decryption of m' results in the plain text.

4. The Communication Model

In order to adopt the above encryption scheme we propose a model for communication which confirms to agent system model given in section 2. First we introduce the concept of Publisher and Consumer agents. Publisher agents belong to the user who wants to publish or put some important data on the web. It can be anything like product information or an important message which the user wants to share with particular group of people. To publish the data, the user first creates two agents P1 and P2. The agent P1 stores the data to be shared and agent P2 contains the secret key of the user. It is extremely important that P1 and P2 reside on two different hosts. This is required because; the last step of our encryption scheme is decryption using the secret key of the user. If a single agent is used then we will be forced to send the secret key with that agent. This secret key then becomes vulnerable for attack by the untrusted host. Before dispatching the agents on the untrusted hosts, UH1 and UH2 respectively, the data in the agent P1 is first encrypted using the public key of the Publisher. This corresponds to the first step of our encryption scheme. Agent P1 also carries with it the location of P2. Thus the Publisher launches its agents P1 and P2 so that they reside on two different untrusted hosts(UH1 and UH2). The Consumers are the users who would like to get the data published by the Publisher. In order to access the data published by the Publisher, the consumer launches its agent C which roams around the network looking for appropriate Publisher agent who can provide it with the required information. When it encounters one it first authenticates itself to the agent and also checks its authenticity. Once the agents authenticate themselves, Publisher agent P1 sends the encrypted message, encrypted using the public key of Publisher, to the agent C. The agent C first reencrypts the encrypted message with the public key of Consumer. This corresponds to the second step of our encryption scheme. Agent C alsoreceives the location of P2 from P1. The location information is used by the agent C to move from the current untrusted host, UH1, to other which is hosting the agent P2, UH2. The agents P2 and C authenticate themselves. After that, agent C sends a doubly encrypted message to P2 which holds the private key of Publisher. P2 in turn decrypts the message once with the private key of the Publisher. This corresponds to the last step of our encryption scheme. The resulting message which is encrypted in the public key of Consumer is sent back by P2 to C. The untrusted platform, UH2 cannot get anything out of this communication as the message is encrypted both ways. Similarly, assuming that the platforms do not collude, it is difficult for the platform UH2 to guess anything out of the secret key stored in P2. Thus the scheme provides total confidentiality, as far as the communication is concerned and maintains data privacy in the agents.

Once the agent gets the encrypted message from P2, it can go back to its home platform, a trusted platform, where it can either store the message in the encrypted form in the database for future reference or it can be decrypted by the host. Figure 3 gives the complete communication model of our scheme. The numbers on the edges gives the sequence of operations performed. Thus, 1 and 2 indicates that the agents P1 and P2 are launched by the Publisher. Similarly, 3 corresponds to launch of Consumer agent C. The order of operations corresponds to the sequence numbers. As such, sequence 3 cannot come before 1 and 2.

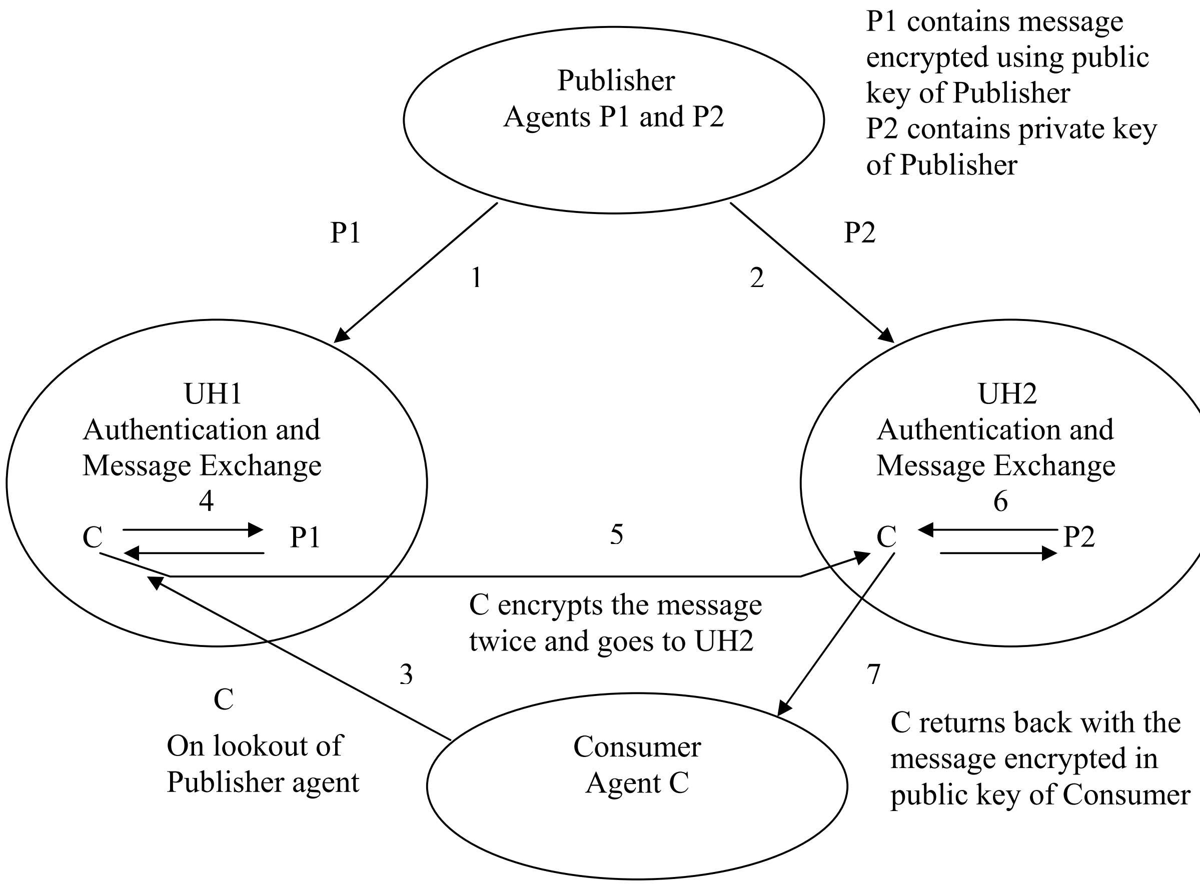

Publisher
Agents P1 and P2
P1 contains message encrypted using public key of Publisher
P2 contains private key of Publisher
P1
1
2
P2
UH1
Authentication and Message Exchange
4
C
P1
5
UH2
Authentication and Message Exchange
6
C
P2
C encrypts the message twice and goes to UH2
3
7
C
On lookout of Publisher agent
Consumer Agent C
C returns back with the message encrypted in public key of Consumer

Fig. 3 The communication model for Publisher and Consumer

The algorithms for Publisher and Consumer using java notations of IBM Aglets platform can be thus summarized as follows:

```
Algorithm_Publisher
{
//Create agent P1 attach an encrypted message and address //of P2 to it and sends it to an untrusted platform specified //by the URL1 in URL List

Slave.create(getCodeBase(),SlaveClassName,
getAgletContext(),  this, getURLList(), new String());

//Create agent P2 attach a private key of Publisher to it and //send it to an untrusted platform specified by the URL2 in //URL List

Slave.create(getCodeBase(),SlaveClassName,
getAgletContext(),  this, getURLList(), new String());

}
```

```
// P1 waits for the Consumer Agent to send a request //message to it. If message received, it can handle it using //simple statement like:

public boolean handleMessage(Message msg) {
      if (msg.sameKind("request")){
// authenticate the Consumer agent
// on successful authentication send the request grant
   message
// on receiving ready message from Consumer agent, send
   the stored encrypted message and address of P2 to it
   }
}
Algorithm_Consumer
{
 // Consumer creates an agent C1 containing required
//  preferences of  Publisher and request and sends it to an
//  untrusted platform specified by the URL1 in URL List
Slave.create(getCodeBase(),SlaveClassName,
getAgletContext(),  this, getURLList(), new String());

//  On receiving request grant from P1, authenticate P1
// send ready message to P1
//receive encrypted message and reencrypt it using its
public key
// extract URL2 of P2 from the message and move to
URL2
// authenticate itself as above to P2
// send the doubly encrypted message to P2
// get back the message which is once decrypted using
the private key of Publisher, contained in P2 (or the
message directly encrypted in public key of Consumer)
// get back to the home platform
}
```

4.1 Using Homomorphic Properties of ElGamal Encryption

An additional property of E-E-D is the ability to add two different but similarly encrypted messages while preserving confidentiality. When it is possible to compute E(m1 + m2) from E(m1) and E(m2) without decrypting any of these values, the encryption algorithm is denoted as being homomorphic in addition. This is possible in ElGamal given that the security parameter k is equal for E(m1) and E(m2). In that case, if two messages m1 and m2 are encrypted using an equal k, the ciphertexts will look as follows:
$\gamma 1 = \alpha^k \bmod p$; $\delta 1 = m1y^k \bmod p$
$\gamma 2 = \alpha^k \bmod p$; $\delta 2 = m2y^k \bmod p$
where y is the public key. Note that here δ2 is computed using the original ElGamal encryption scheme [3] and not by the E-E-D scheme. Adding δ1 and δ2 gives $\delta 1 + \delta 2 = (m1 + m2)y^k \bmod p$, which is equal to the direct encryption of m3 where m3 = m1 + m2, hence it is possible to add two numbers without having to decrypt one of the messages.
One of the conditions for using ElGamal is randomly choosing a new security parameter k for each encryption. Only in certain cases it is allowed to use one k twice. When one message is encrypted twice, separately, using an equal k will result in two equal ciphertexts. Furthermore, given two ciphertexts and equal values for k, it is possible to derive the ratio of the plaintexts i.e.($\delta 1/\delta 2 = m1/m2 \bmod p$). Two parties that each encrypt a message use equal values for k and the public key. Then both parties can compute the other party's plaintext without knowing the corresponding private key. Taking these risks into account, ElGamal encryption can only be used with equal values for k when either the encrypting parties are one single entity or when the encrypting parties fully trust each other, e.g. when protection is only necessary towards a third party. The use of homomorphic property of ElGamal scheme is desribed in the next section.

4.2 A Model with Multiple Agents for Voting Application

The figure 3 below depicts the general model for a data collecting agent T which collects data from multiple agents, A and B and performs operation on that data. In this case, the security requirements are to provide confidentiality when the data is collected and the ability to perform an operation on this collected data. Homomorphic addition on the collected data is performed, as was described in the previous section to preserve confidentiality during the operation at an untrusted host. In the figure below UH and TH indicates Untrusted and Trusted hosts respectively. The order of operations is indicated by the roman numerals.
A user wishes to know exactly how many other users are interested in talking with him. Such a scenario can be useful in applications such as voting where it is only required to count the number of votes in favor of a given user, X. Hence the user agent S on its own trusted platform THt launches another agent T which roams around in the network collecting information from all the agents interested in voting the user. This operation is not shown in the above figure to avoid complication. As was explained in previous model the different users (voters in this context) A and B communicate with T using our EED scheme by launching two agents each A1,A2, B1 and B2. In this model it is assumed that the agents A1 and B1 has certain priority set in their messages regarding the agents with which they wish to communicate (vote). For the willingness the message can be 1 or else it is 0. Thus, agent T receives either 1 or 0 as a message from other agents. Moreover these messages are encrypted on the fly using our EED scheme described previously. Now the agent T can collect all the messages so received from other agents and perform the homomorphic addition of messages at any other untrusted host (UH5) as shown in the figure 4. After performing the addition it can return back to its parent agent S which then decrypts the result to know exactly how many users voted for it.

5 Conclusion

It is a major challenge to provide secured Mobile Agent communication. The basic problems of code mobility restrict the use of conventional security measures to be adopted directly in this context. There is an urgent need for new effective and efficient solutions in this area. We have explicitly tried to provide data privacy to an agent. We have proposed a secured communication model based on our EED scheme. We have mathematically proved the scheme and used it to ensure that the data possessed by the agents is secured at all times when it is executing at any of the untrusted hosts. Similarly, we have also shown how the homomorphic property of the ElGamal scheme can be incorporated with our model to realize a typical secured web application like voting. We are currently into the implementation phase of the scheme. One important aspect that needs to be taken care off is the agent authentication mechanism and ability to detect tampering of agent data. In our scheme we have assumed that the standard mechanism of blind digital signatures is in place. Our future work includes proposing a suitable digital signature technique using ElGamal scheme, which happens to be our base system, for the model.

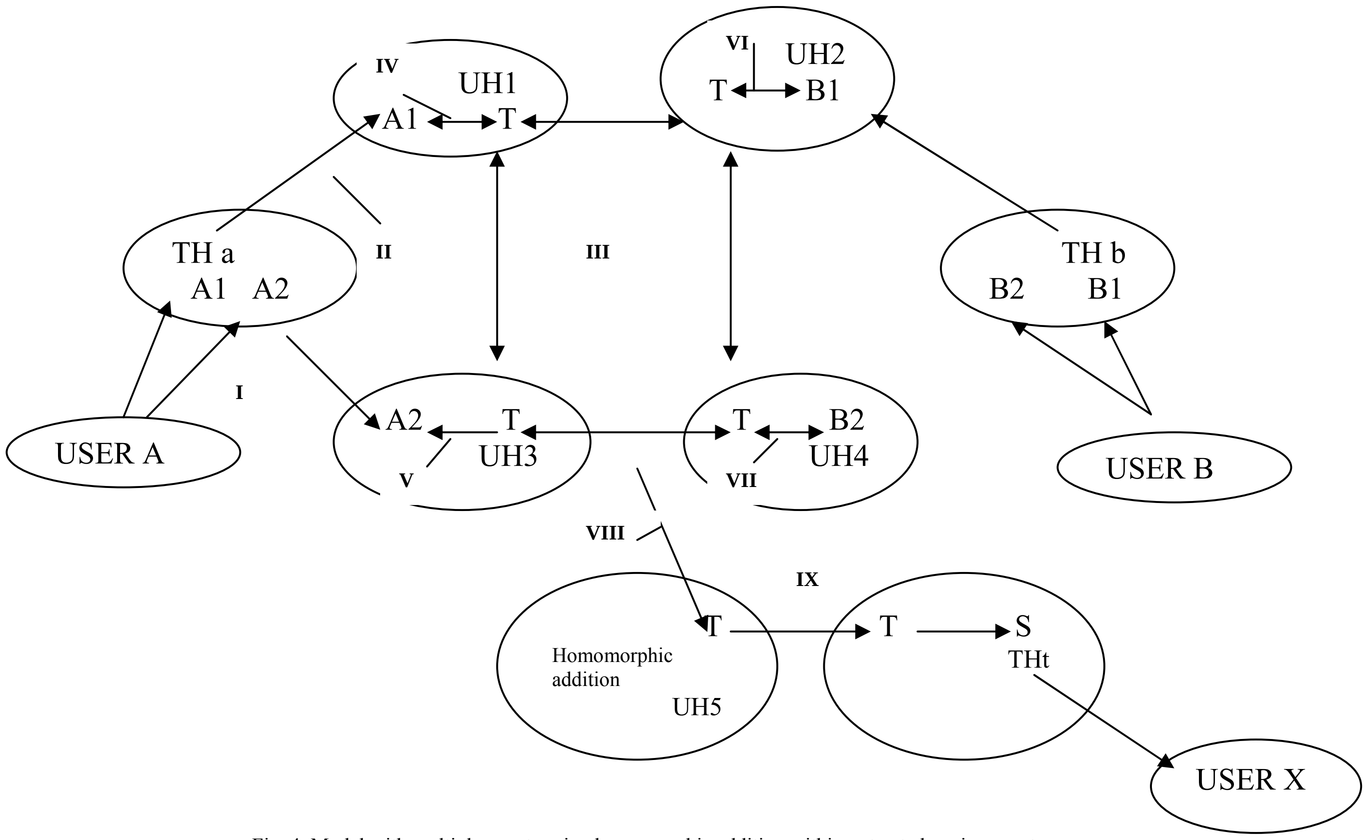


Fig. 4 Model with multiple agents using homomorphic addition within untrusted environments.

References

[1] Mobile Agent Security, National Institute of Standards and Technology, Special Publication 800-19, August 1999. Wayne Jansen and Tom Karygiannis. http://csrc.nist.gov/publications/nistpubs/800-19/sp800-19.pdf

[2] Countermeasures for Mobile Agent Security, Computer Communications, Special Issue on Advanced Security Techniques for Network Protection, Elsevier Science BV, November 2000. Wayne Jansen. http://csrc.nist.gov/groups/SNS/mobile_security/documents/mobile_agents/ppcounterMeas.pdf

[3] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–72, 1985.

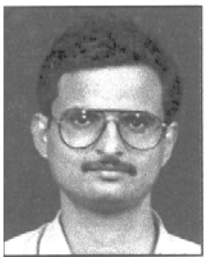
[4] D.S.Adane, S.R.Sathe, P.D.Adane,'Data Privacy in Mobile Agent Communication', 3rd IEEE/IFIP International Conference in Central Asia on Internet, ICI 2007. Tashkent, 26-28 Sept. 2007.

[5] Agent Management, FIPA '97 Specification, part 1, version 2.0, Foundation for Intelligent Physical Agents, October 1998. <URL: http://www.fipa.org/spec/FIPA97.html >

[6] Mobile Agent System Interoperability Facilities Specification, Object Management Group (OMG) Technical Committee (TC) Document orbos/97-10-05, November 1997. <URL: http://www.omg.org/techprocess/meetings/schedule/Technology_Adoptions.html#tbl_MOF_Specification>

[7] J. E. White, Mobile Agents, in J. M. Bradshaw (Ed.) Software Agents, AAAI/The MIT Press, 1997.

D.S.Adane received M.Tech degree in Computer Science and Engineering from I.I.T Guwahati. He is currently doing his PhD in Computer Science and Engineering from Visvesvaraya National Institute of Technology, Nagpur and working as Senior faculty in Information Technology Department, Shri Ramdeobaba Kamala Nehru Engineering college, Nagpur. His research interests include Distributed and Mobile Computing and Mobile Agents. He is a Life Member of ISTE and a Member of Institution of Engineers India (MIE).



S.R. Sathe received M.Tech. in Computer Science from I.I.T. Bombay and Ph.D. from Nagpur University. He is currently working as Asstt. Prof. in Computer Science and Engineering Department at Visvesvaraya National Institute of Technology, Nagpur. His research interests include Parallel and Distributed Systems, Mobile Computing and Algorithms.