National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC-CNRC

*Anonymous Communication for Mobile Agents**

Korba, L., Song, R. and Yee, G.
October 2002

* published in: 4th International Workshop on Mobile Agents for Telecommunication Applications MATA'02 Barcelona, Spain. October 22-24, 2002. NRC 44948.

Canada

Anonymous Communications for Mobile Agents[1]

Larry Korba, Ronggong Song, and George Yee

Institute for Information Technology
National Research Council of Canada
Ottawa, Ontario K1A 0R6, Canada
{Larry.Korba, Ronggong.Song, George.Yee}@nrc.ca

Abstract. Anonymous communication techniques are vital for some types of e-commerce applications. There have been several different approaches developed for providing anonymous communication over the Internet. In this paper, we review key techniques for anonymous communication and describe an alternate anonymous networking approach based on agile agents intended to provide anonymous communication protection for mobile agent systems.

1 Introduction

Mobile agent systems have been identified as a programming paradigm that allows flexible structuring of distributed computation over the Internet [1]. It is expected that they will take an important role in the future information society and especially in e-commerce applications. However, security and privacy protection for mobile agents is still a young discipline. Privacy issues in particular are raised by user demand and government regulations. One cause for concern is the fact that agents may be operating autonomously; this could present a significant threat to privacy due to the wealth of personal information in their processing or under their control.

In this paper we describe an alternate Onion Routing (OR) approach to provide anonymous communications for mobile agents, protecting their communications against traffic analysis. Our principles are as follows. First, our approach is implemented in the application layer, and the onion nodes are implemented using agile onion agents in order to provide rapid deployment for an anonymous data forwarding service for mobile agents. The onion agents communicate with one another via the Agent Communication Language (ACL). Second, the Onion Routing network consists of a large set of onion agents. The initial communication between any application agents is first passed to an onion proxy agent. The onion proxy agent then chooses several random onion agents, to make a random route, it then encrypts the data using a nested encryption algorithm, and sends the encrypted data through to the next node in its route. The Route is dynamically arranged making it difficult to compromise the data. As well, agile OR agents may be dispatched to different nodes for further robustness.

The rest of the paper is organized as follows. The issues of traffic analysis related to agent-based applications are briefly discussed in the next section. In Section 3, some existing approaches for protecting anonymous communication networks against traffic analysis are reviewed. In Section 4, a dynamic, ad hoc Onion Routing approach based on agile agents is described in detail. In Section 5, some concluding remarks and directions are presented for further research.

---

[1] NRC Paper number: NRC-44948

2 Problem Statement

Traffic analysis is a serious menace to agent-based applications. An adversary can monitor and compromise certain parts of a distributed agent system by matching a message sender with the receiver. Protecting the disclosure of communication partners or the nature of communication between partners is an important requirement for confidentiality in an e-business context. It is also a property desired by agent users who want to keep their agent lives and relationships private. On the other hand, since most agent platforms use global name service in order to provide global tracking service, it makes traffic analysis attacks simple. The major attacks are described as follows.

- ***Communication Pattern Attack:*** An adversary may discover considerable useful information simply by tracking the communication patterns when agents send and receive messages.
- ***Message Coding Attack:*** An adversary can easily link and trace some messages if the messages do not change their coding during transmission.
- ***Timing Attack:*** An adversary can observe the set of messages coming into the network and the set of messages going out of it, to obtain some useful route timing information by correlating the messages in the two sets.
- ***Packet Volume Attack:*** An adversary can observe the amount of transmitted data (e.g. the message length, number of messages).
- ***Metadata Attack:*** An adversary can find the identity of an agent from metadata even if the data itself is not accessed in any way.
- ***Message Delaying:*** The adversary can delay messages to obtain some information regarding how data is handled within a communication network.
- ***Intersection Attack:*** An adversary may trace some agents by observation over a long period searching for special distinguishable behavior.
- ***Collusion Attack:*** A corrupt coalition of agents or parts of the system may be able to trace some agents.
- ***Denial of Service Attack:*** An adversary may obtain some information about the routes used by certain agents by rendering some nodes inoperative.
- ***Replay Attack:*** An adversary, who observes the incoming and outgoing messages, would capture and replay a message to the node to try to take it over.

Privacy for e-commerce has been recognized as a vital requirement for many years. However, TCP over IP version 4 is designed to allow computers to easily interconnect and to assure that network connections will be maintained even when various links may be damaged. This same versatility makes it rather easy to compromise data privacy in networked applications. For instance, networks may be sniffed for unencrypted packets, threatening the confidentiality of data, or using the attacks listed above, wherein the nature of a communication or information about the communicators may be determined. Research has led to techniques that provide varying levels of private communication between parties. The next section describes some of the more commonly known network privacy technologies concisely.

3 Existing Approaches for Anonymous Networks

(1) MIX-Network

In order to enable unobservable communication between users of the Internet, David Chaum [2] introduced MIX-networks in 1981. A MIX network takes a list of values as input, and outputs a permuted list of function evaluations of the input items, without revealing the relationship between input and output elements.

A MIX-network is composed of MIX nodes. A MIX node is a processor that receives a certain number of messages, modifies them using some cryptographic transformation and outputs them in a random order in such a way that one cannot correlate messages that "come in" with messages that "go out". MIX nodes can be used to prevent traffic analysis in roughly the following manner.

(1) The message will be sent through a series of MIX nodes, say $i_1, i_2, \ldots, i_d$. The user encrypts the message with an encryption key for node i_d, encrypts the result with the key from node i_{d-1} and so on with the remaining keys.
(2) The MIX nodes receive a certain number of these messages, which they decrypt, randomly reorder and send to the next nodes in the routes.

Each MIX node in the network knows only the previous and next node in a received message's route. Hence, unless the route only goes through a single node, compromising a MIX node doesn't trivially enable an attacker to violate sender-recipient privacy. When using only one MIX, one must rely upon security of that node completely. Usually several MIXes are used in a chain. In this manner, any single MIX does not have enough information needed to reveal communication relations. At worst, a MIX may only know either sender or receiver.

(2) Onion Routing

The primary goal of Onion Routing [3, 4, 5] is to provide strongly anonymous communications in real time over a public network with reasonable cost and efficiency. A secondary goal is to provide anonymity to the sender and receiver, so that the responder may receive messages but be unable to identify the sender, even though the responder may be able to reply to those messages.

In onion routing, initiating applications make connections through a sequence of onion routers instead of making socket connections directly to responding machine. Onion routers are computer programs that perform application-layer routing for the network. The onion routing network allows an anonymous connection between the initiator and responder. Onion Routing builds anonymous connections within a network of onion routers, which are, roughly, real-time Chaum MIXes. While Chaum's MIXes could store messages for an indefinite amount of time while waiting to receive an adequate number of messages to mix together, a core onion router is designed to pass information in real time, which limits mixing and potentially weakens the protection. Just as large volumes of traffic improve the protection of real-time MIXes, large traffic is vital to strengthen onion router networks.

Onion routers in the network are connected by longstanding socket connections. Anonymous connections through the application layer onion routing network are

multiplexed over these longstanding connections. For any anonymous connection, the sequence of onion routers in a route is strictly defined at connection setup. However, each onion router can only identify the previous and next hops along a route. Data passed along the anonymous connection appears differently at each onion router, so data cannot be tracked en route.

4 Our Approach (Ad hoc Onion Routing)

Our ad hoc Onion Routing network is designed to provide anonymous communication for multiple mobile agents. It is implemented using the JADE multi-agent platform. Each JADE platform has several onion agents which provide an anonymous data forwarding service, and at least one onion monitor agent which keeps track of the location of the onion agents. The network is dynamically set up using mobile onion agents. To do this, onion agents migrate across the network to different network nodes where the Jade platform is running in order to maximize the number of onion agents on the network, thereby making the private communication service more effective. In Section 5, we discuss another situation wherein agent mobility enables a highly dynamic anonymous network.

4.1 Terminology and Notations

Notations used in the paper are defined as follows.

- **AMA:** The Application Message Agent is an application agent that makes anonymous connections to the ad hoc onion routing network. The sole purpose of the agent is to test and demonstrate the ad hoc onion routing network.
- **ONA:** The Onion Node Agent acts as both a proxy to the onion network and as an onion router. As a proxy, the ONA can perform either initiator proxy function or responder proxy function. As the initiator proxy, the ONA responds to a "request" message sent by the initiator AMA, creates an onion and encrypts the data using a nested encryption algorithm. For the responder proxy, the ONA decrypts the last layer of onion and data payload, and forwards the data to the responder AMA. As an onion router, the ONA decrypts one layer of onion and data payload and forwards them to the next ONA. We use ONA_i as the address of the i-th ONA in the ad hoc onion routing network.
- **OMA:** The Onion Monitor Agent facilitates onion routing by monitoring ONAs to keep track of their location. Every platform will have several ONAs and a single OMA. Upon start up, the OMA searches for all ONAs on its platform and other OMAs located on other platforms. After an OMA has the location of agents currently available it is able to create a layout of the onion routing network (by making an ONA list). The OMA can then pass an ONA list to other OMAs.
- **AOT:** The Anonymous Onion Tunnel is an anonymous tunnel between the initiator proxy and the responder proxy. It is composed of anonymous tunnel segments between ONAs.
- **OTI:** The Onion Tunnel Index is a random value used in combination with the destination agent address to identify the anonymous tunnel segment between two ONAs.
- $E_{PK_i}(K_i)$**:** The symmetrical key K_i is encrypted with the ONA_i's public key PK_i, e.g. *RSA*.
- $E_{K_i}(\boldsymbol{M})$**:** The message M is encrypted with the symmetrical key K_i, *e.g. DES.*
- *H (M*): The message M is hashed with a hash function*, e.g. MD5.*

4.2 Dynamic, Ad hoc Onion Routing Topology

The ad hoc Onion Routing network consists of many multi-agent platforms. Each agent platform has several ONAs and a single OMA. The ONAs can be located in different containers. The OMA usually is located in the main container. The ONAs connect to each other via ACLMessage [6]. The OMAs communicate to each other via a multicast mechanism. Every ONA accepts the data stream from its customer application agents or other ONAs, and forwards the data stream to the next ONA according to the routing information. An anonymous routing protocol would be a desired approach for this system. At present, the initiator onion proxy randomly picks several ONAs to make an anonymous route. The ad hoc onion routing topology is illustrated in Fig. 1.

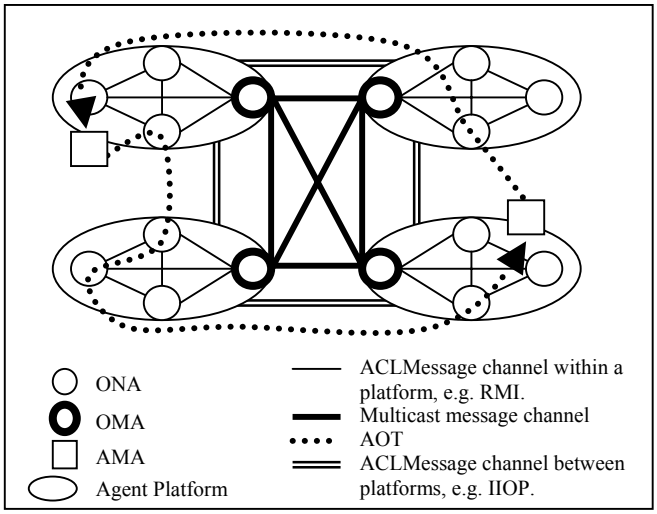

Fig. 1. Ad hoc Onion Routing Topology.

The ad hoc onion routing network allows the connection and communication between the application agents in a manner that allows them to remain anonymous. The anonymous connections hide information that might reveal who is connected to whom, and for what purpose.

To begin an anonymous session, the initiator application agent sends its request message to the ONA that was registered to act on its behalf using a secure connection. We call the ONA an initiator onion proxy for the initiator application agent. According to the destination application agent address, the initiator ONA randomly picks several ONAs to form the anonymous route, and encrypts the original communication data using the nested encryption algorithm described in the Section 4.4. The process of creating the onion not only protects the payload, but also distributes the symmetric keys. The ONA then encapsulates the encryption data payload using the ACLMessage and sends it to the next ONA. Finally, the original communication data is forwarded to the responder application agent. In addition, the expiration time of each anonymous onion channel can be set up according to the privacy protection requirements or speed requirements. For instance, in cases where information must be protected in the strictest sense, the route taken may change with each message exchange. In cases

where performance is an issue, the expiration time may be lengthened to minimize the overhead incurred with a change of route.

4.3 Dynamic, Ad hoc Onion Routing Protocols

To provide anonymous communication between application agents, the ad hoc onion routing network includes three protocols: onion creation protocol, data transmission protocol and destroy protocol as follows.

(1) Onion Creation Protocol

In our ad hoc onion routing, only the initiator onion proxy knows the AOT, and other ONAs only know the previous and next ONAs that form the AOT. Thus, the initiator proxy must be a trusted ONA for the initiator. The onion creation protocol is described as follows.

① The initiator ONA first randomly picks a series of ONAs forming a route through the onion routing network, and constructs an onion creation signal according to the responder agent address of the initiator's ACLMessage. An OMA is required in order to keep track of the ONAs' location. Assuming the route consists of ONA_1, ONA_2, …, ONA_n where ONA_1 is the initiator proxy, ONA_n is the responder proxy, and $OTI_{i,j} = H(ONA_i, ONA_j, K_j, Random\ Number)$, Fig. 2 depicts the onion creation signal (using ONA_i to mean an address).

$$
\begin{aligned}
&E_{PK_2}(K_2)\ E_{K_2}(\mathrm{ONA_3}, \mathrm{ONA_1}, \mathrm{OTI_{1,2}}, \mathrm{OTI_{2,3}}, \text{Exp-Time},\\
&\quad E_{PK_3}(K_3)\ E_{K_3}(\mathrm{ONA_4}, \mathrm{ONA_2}, \mathrm{OTI_{2,3}}, \mathrm{OTI_{3,4}}, \text{Exp -Time},\\
&\qquad \cdots\cdots\\
&\qquad\quad E_{PK_{n-1}}(K_{n-1})\ E_{K_{n-1}}(\mathrm{ONA_n}, \mathrm{ONA_{n-2}}, \mathrm{OTI_{n-2,n-1}}, \mathrm{OTI_{n-1,n}}, \text{Exp-Time},\\
&\qquad\qquad E_{PK_n}(K_n)\ E_{K_n}(\mathrm{NULL}, \mathrm{ONA_{n-1}}, \mathrm{OTI_{n-1,n}}, \mathrm{NULL},\\
&\qquad\qquad\qquad \textit{Responder Agent Address,}\ \text{Exp-Time}))\ldots)
\end{aligned}
$$

Fig. 2. Onion Creation Signal.

② The initiator ONA then sends the above onion creation signal to ONA_2, and stores K_n, K_{n-1}, …, K_2 as the nested encryption keys for the forward data stream, and the nested decryption keys for the backward data stream.

③ ONA_2 decrypts one layer of the onion creation signal using its private key SK_2, and stores the $OTI_{1,2}$, $OTI_{2,3}$ and symmetric key (K_2). Finally, ONA_2 creates a bi-directional connection between $OTI_{1,2}$ and $OTI_{2,3}$, and uses K_2 as the decryption key for the forward data stream and the encryption key for the backward data stream over the connection. All intermediate ONAs behave as in this step.

④ Finally, ONA_n decrypts the onion creation signal using its private key SK_n, and stores K_n as the decryption key for the forward data stream and the encryption key for the backward data stream.

The AOT between the initiator ONA and the responder ONA is established using the protocol described above. Fig. 1 depicts the AOT.

(2) Data Transmission Protocol

Based on whether the initiator agent hides its name and address from the responder or not, two options are available for providing different degrees of anonymous communications. The initiator can choose the best one according to its requirements for privacy.

The first option is that the initiator hides its name and address from any entities except for the initiator ONA. In this situation, the initiator will have a higher degree of anonymous protection. The second option is that the initiator doesn't hide its name and address from the responder. The technique is described as follows.

① The initiator agent first prepares its ACLMessage according to its desired degree of anonymity, i.e. putting its name and address in the ACLMessage or not, and then sends the ACLMessage to the initiator ONA, which is a locally registered ONA or a registered ONA in the initiator's home platform, using security channel $OTI_{I,1}$.

② The initiator ONA verifies whether or not the initiator is its registration customer, and gets the responder agent name and address from the ACLMessage. The initiator ONA then creates an AOT using the above onion creation protocol to make a connection between the $OTI_{I,1}$ and the $OTI_{1,2}$, and encrypts the whole data using the nested encryption algorithm, i.e. the data first is encrypted with K_n, *then* K_{n-1}, *..., and final* K_2. Finally, the initiator ONA creates a new ACLMessage for the encrypted data payload with $OTI_{1,2}$ and sends the new ACLMessage to ONA_2.

③ ONA_2 decrypts one layer of the encrypted data payload using its decryption key K_2, and then creates a new ACLMessage for the data payload, and sends the new ACLMessage to the next ONA according to the $OTI_{2,3}$.

④ All intermediate ONAs behave as in step ③.

⑤ Finally, ONA_n decrypts the last encrypted layer using its decryption key K_n, to get the original data. ONA_n then sends the original data to the responder using security channel $OTI_{n,R}$.

When a backward data stream is sent from the responder, the inverse processing to the above is performed, except that the cryptographic operation is an encryption operation for each ONA except for the initiator ONA node. The initiator ONA decrypts the backward data stream with K_2, then K_3, ..., and finally K_n.

(3) Destroy Protocol

Every AOT can optionally have an expiration time. An AOT will be destroyed immediately after a destroy signal is sent, or when its expiration time has expired. A destroy AOT signal can be made and sent by the initiator ONA, responder ONA and any ONAs in the AOT. There are several situations that would lead to destroying an AOT as follows.

- The first situation is that the AOT has an expiration time and it has expired. Thus, all ONAs automatically destroy the AOT according to the expiration time.
- The second situation is that either the initiator or responder sends a destroy signal for some reason (e.g. a session ends). Thus, the initiator ONA or responder ONA will create a destroy AOT signal, and send it to the next ONA. The next ONA finds its next OTI, and then creates a new destroy AOT signal, sending it to its next ONA, and so on, eventually destroying the AOT.
- The third situation is that any intermediate ONA sends a destroy AOT signal for

some reason. The ONA will create two destroy AOT signals, sending them to the next ONA along the two directions, which will eventually destroy the AOT.

- The final situation is that the AOT doesn't have an expiration time. In this situation, the AOT is a one-time AOT, i.e. the data payload accompanies the onion without any need for AOT persistence. After transmission of the onion, the AOT is destroyed automatically.

4.4 Dynamic, Ad hoc Onion Routing Implementation

In this section we provide a description of our prototype implementation. All components are coded in Java. We used JADE 2.5 [6] to provide a framework for software agent development, and IAIK-JCE 3.0 as the cryptographic package. Due to space limitations, only a very concise description is given below.

(1) Application Message Agent

The AMA represents the agent application that may wish to use our ad hoc onion network infrastructure. For our prototype the application is a simple version of popular instant message chat programs. Two AMAs may simply exchange text messages between each other. By requesting proxy services from an ONA within the ad hoc onion routing network, a pair of AMA agents may make their communications private. Fig. 3 depicts the design structure used to create the message agent application.

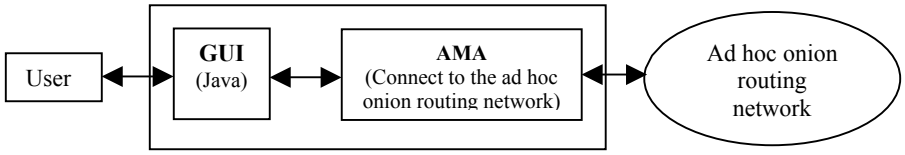

Fig. 3. Structure of the application message agent.

AMA receives the text messages from the user, makes a standard data structure and sends it to its registered ONA. Fig. 4 depicts the standard data structure for the text message.

Version/Protocol	Initiator Address	Responder Address	Message

Fig. 4. This figure outlines the fields within the standard data structure.

(2) Onion Node Agent

An ONA has the functionality of both an onion proxy and an onion router. There are two sets of input/output parameters for the ONA: (i) as a proxy, and (ii) as a router.

As an initiator proxy, the ONA first receives the standard data structure from an AMA. It then makes an onion creation signal described in Fig. 2. It also creates a nested encryption data payload as shown in Fig. 5, and sends both the onion and the encrypted payload to the next ONA.

$OTI_{1,2}$	$E_{K_2}(E_{K_3}(\ldots(E_{K_{n-1}}(E_{K_n}($ *Responder address*, Data$)))\ldots))$

Fig. 5. Nested encryption data payload.

The responder proxy (the ONA connected to the destination AMA), receives the onion and payload from a second-last ONA, and sends the data message to the responder according to the responder address.

As a router, each ONA decrypts one layer of onion and data payload using its private key and symmetric key for the forward data stream, and adds a random padding to the onion structure in order to replace the bits that have been extracted, and uses the next OTI to replace the previous OTI in the OTI field of the data payloads.

In addition, a routing table is built for the ONA according to its previous OTI and next OTI. Two source-destination pairs (one for each direction) are added to the routing table for each ONA when an anonymous onion tunnel is built. Table 1 depicts the routing table in ONA_i.

Table 1. Routing table in ONA_i.

Source	Destination
$OTI_{i-1,\,i}$ (ONA_{i-1})	$OTI_{i,\,i+1}$ (ONA_{i+1})
$OTI_{i,\,i+1}$ (ONA_{i+1})	$OTI_{i-1,\,i}$ (ONA_{i-1})

(3) Onion Monitor Agent

When a platform starts up and wants to join the ad hoc onion network it instantiates an OMA. The OMA communicates with the platform's Directory Facility (DF) and other OMAs while managing an address table of the local ONAs and an address table of the global ONAs. Regarding global activity, the OMA sends a multicast message to other OMAs in the network to identify itself. For local activity, the OMA Registers itself with the DF, searches the DF for all local ONAs, and then pings these agents to check if they are still alive.

4.5 Other Considerations

This is a first-generation prototype to provide an anonymous communication service for mobile agent systems. In order to make the ad hoc onion routing network efficient and more robust, the following requires further research.

- Reliability: The system should provide a mechanism in order to determine if an onion/ payload is successful in reaching its destination. This is particularly an issue for link failure within the AOT. A recovery mechanism is needed to reroute and resend messages that do not reach their destinations.
- Performance: The system needs further testing and profiling to improve performance. The key culprit in producing degraded performance is the use of the public key algorithm. While performance can be improved by giving the route a longer expiration time, doing so also increases the chance that the network may be compromised.
- Routing: The system needs a more effective routing algorithm and a mechanism to maintain the routing tables efficiently for the entire network.
- Scalability: The system should provide good scalability. Currently, the operation of the OMA limits the scalability of our current implementation. A newly created ONA must receive a table containing a list of the other onion nodes operating on the network. For many thousands of nodes, this list would be very large, thus tying up bandwidth when many ONAs start up.

We are examining a variety of approaches to deal with the drawbacks of the current prototype. For example, regarding routing, we have developed approaches for incrementally adding new ONA nodes to the ONA list table. As well, we are developing approaches that would support intelligent route creation: selecting routes based upon criteria specified to by the initiating ONA.

Regarding reliability, we have included in our protocols labeling techniques to manage error recovery in the face of link failure. For the most part this is a matter of replicating the way in which the network layer handles link failures in TCP/IP.

5 Discussions and Conclusions

Mobile and multi-agent systems will play important roles in the future information society, especially for e-commerce applications, in which security and privacy are considered to be the gating factors for their success. Thus security, privacy and trust mechanisms have become the desiderata for mobile and multi-agent applications. This paper described an ad hoc onion routing network that provides data protection against traffic analysis for mobile and/or multi-agent systems. Scalability and performance for our current prototype need further research and development. An interesting area for potential research involves the use of mobile ONAs. In this case, ONAs may transfer from one node to another, while they are exchanging messages. The messages and other state information would migrate with the agents, while they migrate to new locations where the message would be transferred to the next node in the route. This approach would make it more difficult to perform traffic analysis. It also could offer a means for avoiding network nodes that may have been compromised, i.e. taken over by an attacker. This approach leads to management complications. We are investigating the possibilities of extending this idea as well as developing solutions for the other considerations in this new approach for agent network communication privacy.

Acknowledgments

We would like to thank Steven Gao and Chris Dabrowski, who helped implement the first prototype, co-workers Khalil El-Khatib and Yuefei Xu and our IST-EU Fifth Framework Project, Privacy Incorporated Software Agent (PISA), partners [8].

References

1. D.B.Lange and M.Oshima. Seven Good Reasons for Mobile Agents. Communications of the ACM, 42(3), 1999.
2. D.Chaum. Untraceable Electronic Mail, Return Address, and Digital Pseudonyms. Communications of the ACM, vol.24 no.2, pages 84-88, 1981.
3. D.Goldschlag, M.Reed and P.Syverson. Onion Routing for Anonymous and Private Internet Connections. Communication of the ACM, vol.42, no.2, pages 39-41, 1999.
4. D.Goldschlag, M.Reed and P.Syverson. Hiding Routing Information. In R.Anderson, editor, Information Hiding: First International Workshop, Volume 1174 of Lecture Notes in Computer Science, pages 137-150, Springer-Verlag, 1996.
5. M.Reed, P.Syverson and D.Goldschlag. Anonymous Connections and Onion Routing. IEEE Journal on Selected Areas in Communications, vol.16, no.4, pages 482-494, May 1998.
6. JADE -- Java Agent Development Framework. http://sharon.cselt.it/projects/jade/.
7. Institute for Applied Information Processing and Communications Home Page. http://jcewww.iaik.tu-graz.ac.at/.
8. PISA web site: http://pet-pisa.openspace.nl/.