A New Security Model for Collaborative Environments

D. Agarwal, M. Thompson, M. Perry
Lawrence Berkeley Lab
DAAgarwal@lbl.gov, MRThompson@lbl.gov, MPerry@lbl.gov

M. Lorch
Virginia Tech
mlorch@vt.edu

Abstract

Prevalent authentication and authorization models for distributed systems provide for the protection of computer systems and resources from unauthorized use. The rules and policies that drive the access decisions in such systems are typically configured up front and require trust establishment before the systems can be used. This approach does not work well for computer software that moderates human-to-human interaction. This work proposes a new model for trust establishment and management in computer systems supporting collaborative work. The model supports the dynamic addition of new users to a collaboration with very little initial trust placed into their identity and supports the incremental building of trust relationships through endorsements from established collaborators. It also recognizes the strength of a user's authentication when making trust decisions. By mimicking the way humans build trust naturally the model can support a wide variety of usage scenarios. Its particular strength lies in the support for ad-hoc and dynamic collaborations and the ubiquitous access to a Computer Supported Collaboration Workspace (CSCW) system from locations with varying levels of trust and security.

1. Introduction

The fundamental mechanisms available today for authentication to computer mediated resources or communications are public or shared key based credentials (e.g. Kerberos Tickets [9] or X.509 Certificates [5]) , and username/password mechanisms. Authorization, based on the authentication token established by the authentication process, is usually either based on a local access control list or is decided by an authorization server. Most computer software today provides interfaces that interact with the user to perform authentication and make authorization decisions without requiring the immediate intervention of an administrator. In these systems, it is essential that the system be able to gain confidence in the identity of the user and automatically determine authorization.

The current authentication and authorization model described above provides an important function in protecting computer systems and resources from unauthorized use. But, it is not necessarily the correct model for authentication and authorization in computer software that moderates human-to-human interactions (e.g. collaboratory software). In a collaboration, one normally builds trust incrementally through interactions with the other collaborators. If collaboratory software precludes this natural trust building and requires that the trust all be built before the credentials are ever issued, then it will only serve the small percentage of already existing and very solid collaborations. Collaboratory software, to be truly effective, needs to allow for the building of trust between individuals through interactions within the collaboration environment.

Imagine the case where you are at a conference and you sit in on the presentation of a paper that you realize solves a long-standing problem you and your collaborators are dealing with. Ideally you should be able to bring the speaker into the collaboration on the spot, introduce her to your collaborators and begin interacting immediately. At this point you don't trust the person to have access to everything in the collaboration, you also don't have time to contact an administrator and perform verification of identity. These steps should not be necessary at this point. Instead, you should be able to introduce new collaborators to the environment, give them a level of access that identifies that they are some form of a guest in the environment and allows them to interact with the other users in a controlled fashion.

Another example of a scenario typically not supported by prevalent authentication and authorization systems is access using an untrusted client computer. E.g. if you are on travel and the only computer access you have is through an Internet café. You are concerned about entrusting such a computer with your password or your private key but you need to get a very important message to one of your collaborators or attend a very important collaboration meeting. A third common collaboration scenario is creating a new collaboration from scratch, adding a few members at a time. A system that is very easy to set up initially, but that can later be scaled up to something more robust and fuller featured is essential.

In this paper, we describe a framework for authentication and authorization in collaborative environments which supports incremental trust. A first prototype of this model is being implemented in a secure messaging system that is part of the Pervasive Collaborative Computing Environment (PCCE) project at LBNL. The implementation incorporates PKI, username/password, and guest authentication methods. It provides an authorization process that is aware of an individual's trust level. We will use this implementation to study the implications and requirements for support of an incremental trust model. It is our expectation that this model, if successful, will be useful in a broad range of applications beyond collaborative software. Since we are working to prototype this model now, the discussion will leverage off of authentication and authorization mechanisms available today.

2. Requirements and Authentication Model

Most authentication systems require the user to perform two steps. The first is a one-time registration where the user registers and possibly provides credentials to a trusted system representative who approves the user. The second is a per session authentication where the user provides proof that he has access to the registered credential and a secure connection to an agent running with his user-id is established. In our model, we wish to allow various levels of registration with several types of credentials and subsequent per session authentication using one of the credentials that was previously registered. Since in such a model, the user can be authenticated by credentials that may have different levels of trust, the current authenticated level of trust must be considered by the authorization process.

The use cases we want to support include:

- A trusted user connecting in from a machine on which she has her X.509 certificate and private key.
- A trusted user connecting from a machine with his username and password.
- A trusted user connecting from a machine on which he does not have his certificate or password.
- A new user who wants to join a session immediately and to build trust in her identity.
- A new user or guest user who wants to join a single session.
- Several users who wish to spontaneously create a secure collaboration group.

These use cases are representative of both well-established collaborations and ad hoc collaborations. The participants in well established collaborations need a highly trusted way of connecting when they are on a trusted machine and a less restrictive means of authenticating when on travel. In both cases it is of primary importance that there is a low threshold for entering the system. We have considerable experience with systems that are hard to join initially or difficult to reenter. Whenever possible, users will circumvent or refuse to use such systems.

2.1. Registration

In order to support the use cases specified above, the system must provide a registration process and the resulting persistent database of users. The registration process must be flexible enough to allow various levels of registration. The user's registration record may contain one or more authentication credentials which can be X.509 identity certificates, username/password pairs or just usernames. The record also contains the type of registration that was performed. The type of a registration can be elevated by re-registering in a more trusted way. The system should allow subsequent authentication using any registered credential or no credential at all. Varied authentication and registration procedures require that the user's authentication context contains information about the method of registration and session.

To support incremental trust, we will provide three methods of being added to this database: self-registration, trusted user registration, and administrative registration. Self-registration is an on-line process where the user connects to the registration server and provides an official user name, a shorter login id or nickname and optionally one or more credentials (e.g., password, x.509 certificate). This registration is flagged as self-registration. A self-registered user's official name and other information may not be accurate. If the self-registered user has a credential, other users can at least assume that the individual represented by this user name is the same person in each session. A user authenticated without credentials may not be the same person as in previous sessions. A guest user would normally be an example of a non-credentialed user.

Trusted user registration is completed by any of the trusted members of the collaboratory group. The verification process at this point is likely based on direct interaction with the new user or personal knowledge of the new user. The registration information is likely entered by the new user or by the trusted member and is signed off as correct at some level by the trusted member registering the new user. This step may include the addition of a new, stronger authentication credential.

Administrative registration is completed by one or more members of the core collaboratory team. This person should determine in a secure manner that the official user name being used legitimately belongs to the person requesting the registration. Any other information that is being registered such as email address and organization affiliation should also be verified. This step will normally be required to gain the strongest levels of authentication credentials and trust. The registration process will likely be

an incremental process with the users and administrative officials gradually adding information and credentials to the database as the user becomes more trusted in the collaboration.

In summary, persistent registration information consists of some number of the following things:

- A user name (an official name that represents the user),
- A password,
- A X.509 credential (uniquely represented by CA/DN, public key, or CA/serial number)
- An organizational affiliation
- A group affiliation
- The method of registration (self, trusted user, or administrative)

2.2. Authentication

An orthogonal issue in determining trust is how the user authenticated to the current session. A user may have an X.509 certificate that he uses from certain machines, a password that he may use from other sites, and may not even want to type in a password from a very untrusted site (e.g. an Internet café). The user may also not yet have some of the authentication methods available.

As one method of allowing a user to build trust, we will include a "vouched for by" field to be added to authentication information. The intent of this feature is to allow a user who has authenticated with a less secure credential or no credential to be vouched for by one or more better authenticated members who are also on-line. The vouching would take place after a side conversation (either on or off-line) has convinced the trusted member of the authenticity of the untrusted member. The vouched for field is only set for the duration of the session and possibly can be rescinded by the vouching user. It is not yet clear exactly what additional privileges having a vouched for field should convey. We likely do not want vouching for a user to allow the vouched for user to gain more privileges than the vouching user has.

In summary, the authentication information that is saved per session consists of the name (nickname) of the user, the authentication method (certificate, password, null), and, if applicable, one or more “Vouched for by” fields that other users might have added after the user authenticated into the session.

2.3. Authorization

It is likely that new users if they self-register will not be able to register a username/password or a X.509 certificate when they first register. Some collaborations will use the establishment of these credentials as coarse authorization mechanisms; access to particular capabilities might be gated by what authorization mechanism was used to enter the collaboration. In this case, adding a username/password and X.509 certificate to the user’s registration would require the help of a trusted user or administrator.

Alternatively, the system could allow self-registration including all the authentication mechanisms. This would make it easier for the user to provide these credentials but makes it more difficult to perform authorization based purely on the establishment of the credentials. In this case, an authorization service or database would be used to define what level of authorization each user has when they authenticate with different mechanisms. If an authorization server were present, it would also provide a method of performing fine-grained authorization.

2.4. Escort

Another feature that we would like to explore is an ability to provide an escort capability, which would allow a trusted user to escort a less trusted user into an environment where they would not normally be allowed. The idea here is that the less trusted user is only allowed to stay as long as the escort is with them. This idea will likely need to be refined to work in systems that do not revolve around a virtual space metaphor but it is an important capability that is different from the vouched for capability. In the escort capability, the idea is that the less trusted user is continuously supervised by the trusted user.

3. Prototype Secure Chat server

The secure chat tool that is currently under development within the Pervasive Collaborative Computing Environment (PCCE) project at LBNL is providing a prototype implementation environment for investigating this incremental trust model.

Our current implementation is based on a client-server model that supports client and server authentication and encryption of messages exchanged over the network. In order to leverage existing technologies, we modified a public domain IRC server (IRCD hybrid) [6] to replace its TCP sockets with SSL connections. To provide persistence (e.g., unique nicknames and permanent venues) and enhanced presence information independent of any one chat server, we developed a custom PCCE server which also provides authentication and authorization services. Both the IRC and PCCE servers use only SSL network connections and have their own X.509 credentials which are presented to each other and to clients.

Users must pre-register with the PCCE server through either a designated system administrator or a registered user with administrative privileges. After having registered, users log into the system via username and password and can then use the client interface to edit their own personal information and register their X.509

distinguished names. Subsequent login can be by either certificate or username and password, and users who authenticate by certificate are granted extended privileges (e.g, the ability to create new user accounts and permanent venues).

The architecture of the LBNLSecureMessaging system is shown in Figure 1. The asynchronous messaging, presence information, authentication (AuthN), and authorization (AuthZ) services are components of the PCCE (on the left). IRCD, on the right, implements the text-based synchronous chat messages using the IRC protocol. It queries the PCCE server for user authentication and authorization. The client is a graphical user interface shown at the bottom as "Client GUI."

A client starts the LBNLSecureMessaging interface and first establishes an SSL connection to the PCCE server to log into the authentication service, which checks the validity of the identifying information. If the login is successful, the client then connects to the IRC server and sends its authentication information and then the standard IRC Nickname and {Username, Nickname, Host} requests. The IRC server queries the PCCE authentication server over an SSL connection before granting access. If the client is allowed to join the chat facility, all other users are sent the client's presence information and method of authentication.

The client's SSL connections to both servers are kept open to facilitate messaging and notification. All synchronous chat messages go directly to the IRC server which forwards them to targeted recipients through the established connections. The PCCE server sends notifications over established connections to the clients so that users can update their views of the collaboration group as other people join, change authentication and availability status, and leave. In addition to locating collaborators, users can view presence information (including current method of authentication) to help decide how much to trust other users (e.g., whether to initiate or respond to private chat requests or send and reply to notes). This view is shown in Figure 2.

Figure 1 - Diagram of Secure Chat service

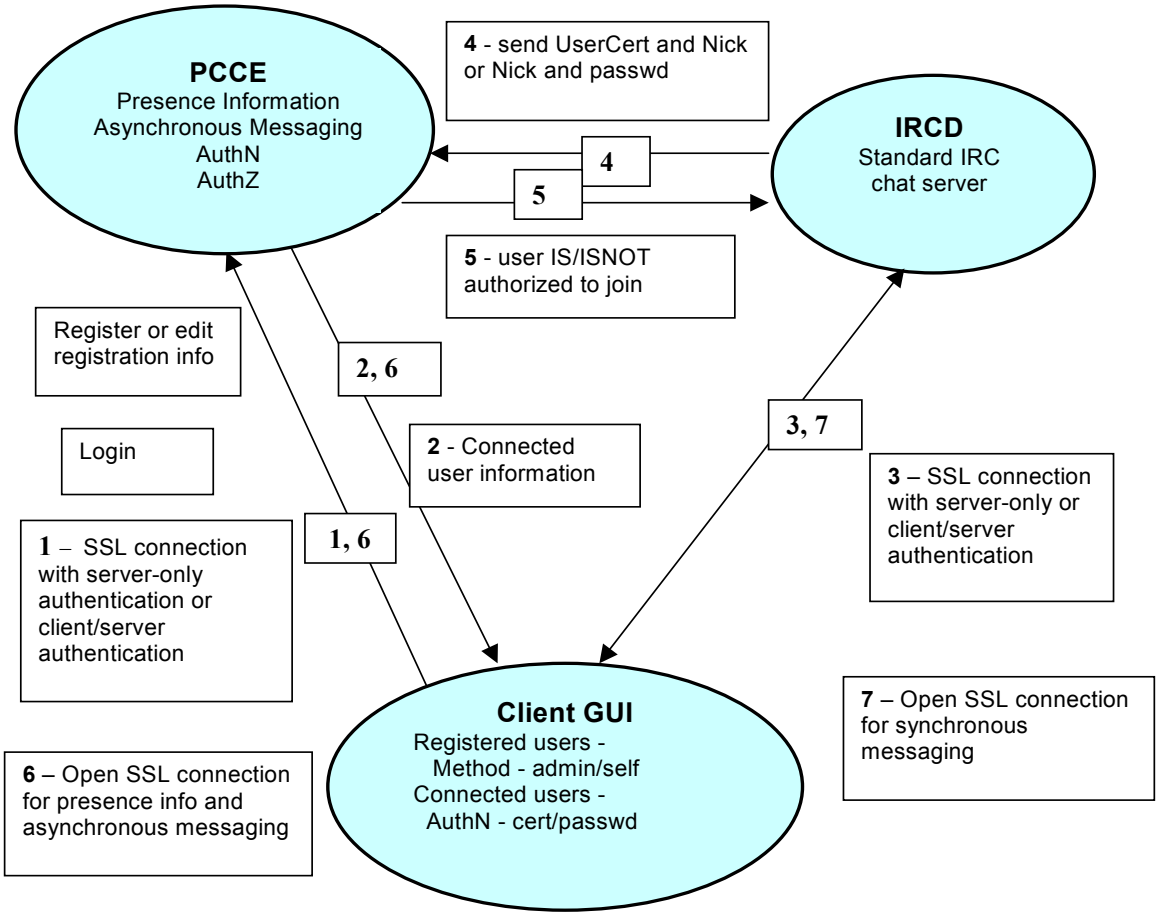
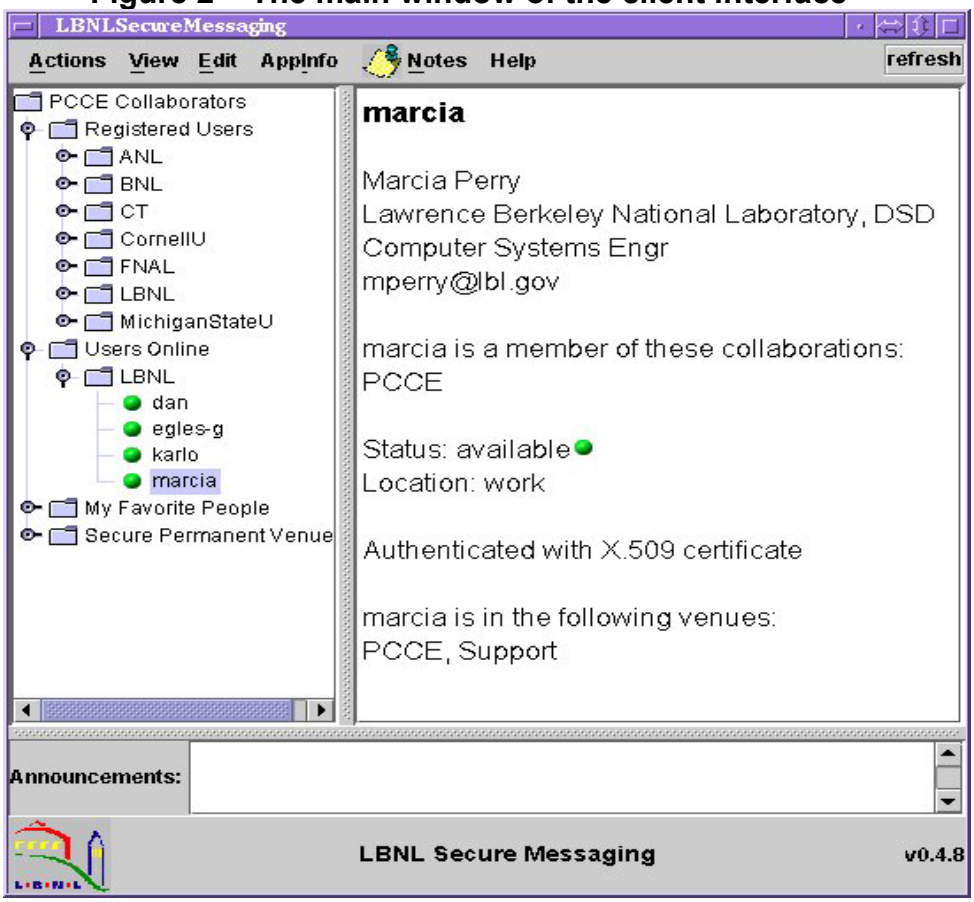


Figure 2 – The main window of the client interface



Users can start a one-to-one conversation with someone. This is a private or invite-only channel that is created when the invitee accepts an invitation from the initiator. Other users may be invited by members in the channel. Any private channel may be made public and any public channel may be made private by the members of the channel. Once a user has joined the chat session the client GUI opens a connection to the PCCE server to find out information about the other users. Personal information currently includes user name, affiliation, job title, and email address. Presence information includes what venues (chat rooms or channels) a user is in and availability status (e.g., available, busy, away, and others we may add later).

There are several aspects of the incremental trust model that still remain to be implemented in the system. For example, allowing a user to escort another user, allowing a user to vouch for another user, self-registration of new users, and guest access. In the current implementation the only access that is controlled is whether a user can join the chat, whether a venue is public or private, and invitations to enter private venues. We plan to add access policy on a per venue level to give finer control than just public/private. For example, some venues might be open only to registered users, but not guests. Individuals may also want to control who can see their presence information. The authentication method of the user as well as identity and other static attributes should be considered as a factor in determining what access and actions a user is allowed.

It is likely that the easiest enforcement mechanism for IRCD authorization is the bot mechanism. Bots are software agents commonly used in IRC systems that are often equipped with administrative rights that reside in a venue (or channel) and can manage the venue's characteristics and enforce the venue's policy. We may need to create bots that sit in the channel and manage it. The bot would then need to interface with the PCCE server or authorization server and enforce authorization actions within the IRCD.

The current implementation of the LBNLSecureMessaging system has approximately 44 registered users and has been in use for the last several months. Our personal experience has been that the multiple authentication mechanisms are very easy and natural to use. The ability to log in with different credentials based on the security of the current location of the user has already proven to be extremely valuable when on travel. In addition, the ability of any trusted member to create new users when logged in with strong

authentication credentials has allowed dynamic inclusion of new collaborators on-demand without waiting for issuance of X.509 credentials or actions by the system administrator. This immediate inclusion of new collaborators is essential to early adoption by new users.

4. Related Work

The concept of trust has been studied extensively by sociologists and psychologists and a great deal is known about how people build and maintain trust. It is clear from this research that trust is an essential ingredient of effective collaborations [7]. Chopra and Wallace [3] have extensively surveyed the literature on social trust and trust in on-line systems, and have produced a set of definitions to characterize trust. They have used these definitions to produce a matrix of classes of electronic environments and the kind of trust that is needed. They note that in on-line relationships previous interactions, introductions and referrals are important ingredients of building trust. Unfortunately, software available today to support collaborative activities does not provide any notion of building trust. A user is either a member of the group and thus trusted or not. Although proposals have been put forward for capturing the trust in a collaborative system such as the one in [8], these systems are generally very complicated and unlikely to be useful in practice. Another approach to building trust has been used in systems such as e-bay which uses a rating and review system [1].

There are a variety of tools available to do on-line authentication. The best known mechanisms include username/password, shared key (e.g. Kerberos[8]), and Public Key Infrastructure (PKI) [15]. Typically, an environment will allow only one style of credential. However, in widely distributed systems, such as Grids, there is increasing interest in supporting several types of credentials. Standard API's such as GSS [11] and protocols such as SAML [16] will support different kinds of credentials, but the software that uses these mechanisms normally only implements one type. For example, while Globus [4] can use GSS implemented with X.509 credentials or Kerberos tickets, the two do not interact. The client and the server must both be using the same type of credential. Since it is an added burden to support different types of credentials, there must be a clear need. We believe that with the increasing reliance on X.509 certificates for strong authentication and the difficulty that people have in obtaining, understanding and managing such credentials, that there is a clearly demonstrated need for multiple styles of authentication to a single system. Most of the strongly secure authentication tokens, such as X.509 certificates, Kerberos or Unix ids, secure id cards or one-time passwords, present a high entry level threshold. Such credentials are essential for long-term access to many resources, but are obstacles to acceptance for any ad-hoc collaborative tool.

There are also many ways to do authorization. The simplest to implement are based on access control lists and flat groups such as are used in Unix. Authorization servers that can be used in conjunction with PKI such as Akenti[18] and the Community Authorization Service (CAS) [13] are also available. A more recent development is the use of third party trusted attribute servers such as Shibboleth [2,17] and Passport [10,14] from which a resource can get a user's attributes or privileges on which to base an access decision. A related idea is the use of a privilege management system such as PRIMA [12] that allows users to selectively use privileges they hold when accessing resources and flexibly delegate privileges for which they are authoritative to their peers.

In our model any type of authorization decision function can be used, but it must not only know the identity of the requestor, but also how he was registered and authenticated since the access policy may be based on such factors.

5. Conclusion

Although there are many mechanisms available for use in securing systems, few of these systems allow the development of trust in an incremental fashion. They generally require that most of the trust be developed before the user ever authenticates into the system for the first time. Existing secure authentication tokens are hard to manage, and the connection mechanisms systems can be quite cumbersome. Sophisticated authorization features that actually implement any differentiation in trust levels between users often go unused because they are non-intuitive and difficult to use.

We believe that the incremental trust model described in this paper defines a system that is easy and intuitive to use. As such, it will fit comfortably into a collaboratory environment and be used to enhance security when needed. Although we have only implemented a small portion of the incremental trust model in our LBNLSecureMessaging collaboration tool, we already see significant benefits in the model.

Aknowledgements

This work was supported in part by the U.S. Department of Energy, Office of Science, Mathematical, Information and Computation Sciences under contract DE-AC03-76SF0098 and by the Virginia Commonwealth Information Security Center (CISC). This paper is also available as LBNL Report number LBNL-52894.

References

1. J. Boyd, "In Community We Trust: Online Security

Communication at eBay," In *Journal of Computer Mediated Communication*, 7(3), April 2003.
2. S. Carmody, "Shibboleth Overview and Requirements," Internet2/MACE Shibboleth Working Group Overview and Requirements Document, February 20, 2001, http://shibboleth.internet2.edu
3. K.Chopra and W.A.Wallace, "Trust in Electronic Environments", *In Proceedings of HCISS-36,* Jan. 2003
4. I. Foster, C. Kesselman, G. Tsudik, S. Tuecke "A Security Architecture for Computational Grids". *IEEE Computer*, 33(12):60-66, 2000.
5. R. Housley, W. Polk, W. Ford, D. Solo "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", RFC3380, http://www.ietf.org/rfc/rfc3380.txt/, 2001
6. IRCD ftp://ftp.blackened.com/pub/irc/ircservers/hybrid/old
7. S. Jarvenpaa and D. Leidner, "Communication and Trust in Global Virtual Teams," In *Journal of Computer Mediated Communication,* 3(4) June 1998.
8. S. Jones and S. Marsh, "Human-Computer-Human Interaction: Trust in CSCW," in the *Special Interest Group on Computer-Human Interaction (SIGCHI) Bulletin,* Vol. 29, No. 3, July 1997.
9. Kerberos: The Network Authentication Protocol", http://web.mit.edu/kerberos/www/.
10. D.P.Kormann and A.D.Rubin, "Risks of the Passport Single Signon Protocol*"* *Computer Networks,* Elsevier Science Press, vol. 33, pp 51-88, 2000 , http://avirubin.com/passport.htm
11. J. Linn "Generic Security Service Application Program Interface, version 2, update 1", IETF RFC2743. Jan 2000
12. M. Lorch, D. Kafura "Supporting Secure Ad-hoc User Collaboration in Grid Environments*", Proc. 3rd Int. Workshop on Grid Computing*, pp 181 – 193, Baltimore, USA, 18 November 2002
13. L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke, "A Community Authorization Service for Group Collaboration," in the *2002 IEEE Workshop on Policies for Distributed Systems and Networks.*
14. Microsoft Corporation ".NET Passport 2.0 technical overview" Microsoft Corporation 06/03/2002
15. PKI Basics: A Technical Perspective, http://www.pkiforum.org/pdfs/PKI_Basics-A_technical_perspective.pdf.
16. SAML "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)" Committee Specification 01, 31 May 2002
17. Internet2 Shibboleth Project, "Shibboleth Architecture " http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-arch-v05.pdf, 2 May 2002
18. M. Thompson, A. Essiari, S. Mudumbai, "Certificate-based Authorization Policy in a PKI Environment," *ACM Transactions on Information and System Security*, Aug 2003