Enabling Secure Ad-hoc Collaboration

Karlo Berket
Lawrence Berkeley National Laboratory
KBerket@lbl.gov

Deb Agarwal
Lawrence Berkeley National Laboratory
DAAgarwal@lbl.gov

Abstract

A secure collaboration environment should be designed to support informal, spontaneous collaborations as well as highly structured environments. Using on-line tools, it should be easy to begin collaborating, and incrementally add users and services as needed. Ideally, the collaboration environment should not depend on any specific resource or server; instead, the resources and servers should add value to the system when they are present. In this paper we describe an approach, built upon secure and reliable multicast communication, that leads to this ideal.

1. Introduction

Many current collaboration tools and environments, such as the Access Grid (AG) [12] provide a set of persistent services to users. However, they often rely on a centralized infrastructure. For example, a user wishing to join an AG venue must connect to a server in order to participate in the collaboration. This architecture works well for highly structured collaborations that can afford to run and administer a highly-available server. However, small collaborations are usually built in an ad-hoc manner and often there is no site available to run the server.

At its core, a collaboration environment depends on the collaborators: (1) being able to reliably communicate with each other and (2) knowing the identities of the other collaborators. When the collaboration is conducted over an untrusted network, such as the Internet, security becomes a critical concern. Providing security allows the legitimate collaborators to feel confident about the identities of their partners and securely communicate with them. Although it is possible to establish the communication among the collaborators using unicast mechanisms (e.g. TCP/SSL), this is very complex, inefficient, and difficult to scale. Instead, secure and reliable multicast is a natural underlying communication layer for a collaboration environment.

An instantiation of secure and reliable multicast communication is provided by a combination of the InterGroup protocols [6] and the Secure Group Layer (SGL) [1]. InterGroup is an extension of the TCP concept to the multi-party case that provides membership services, reliable message delivery, and ordered message delivery. The InterGroup protocols are designed to scale to wide-area environments such as the Internet. SGL is an extension of the SSL concept to the multi-party case that provides the security services required by applications utilizing reliable multicast communication (e.g. InterGroup) in wide-area environments. SGL establishes secure multicast channels among application components. An SGL secure multicast communication channel is established by first exchanging a session key among the legitimate application components. This key is then used to achieve multicast message confidentiality and/or multicast data integrity within the group.

We provide two case studies of collaborative tools that will make use of InterGroup and SGL as core communication services in the collaboration environment. One incorporates these communication services into its design from the beginning. The other started out as a server-based system that is migrating to using InterGroup and SGL.

The first is *scishare*, [13] an information-sharing and discovery system. *Scishare* enables scientists to store and manage information on local storage facilities while sharing it with remote participants. This system is designed from the ground up as a collaboration tool built upon the principal of secure ad-hoc collaboration.

The second is the Pervasive Collaborative Computing Environment (PCCE). [3] The goal of the PCCE is to provide collaboration tools that provide the feeling of working together on an ad hoc or continuous basis. The PCCE system currently relies on a server to coordinate collaborator activities. This server keeps track of the set of authorized users, the set of users currently participating in the collaboration, and the tools available in the environment. The current design forces users of PCCE to

rely on this server. PCCE is migrating to using an InterGroup and SGL core for communication so that the collaboration can operate without the server. By removing the dependence on the server, PCCE gains the ability to run in a purely ad-hoc manner. The PCCE server, if present, will enhance the functionality of the collaboration, rather than creating a (security) bottleneck.

The rest of this paper is laid out as follows. In Section 2 we take a closer look at the issues involved in enabling secure ad-hoc collaboration and present solutions to some of the arising problems. In Section 3 we present how these solutions are being put into practice by examining the *scishare* information sharing tool and the PCCE. Finally, in Section 4 we make a number of concluding remarks.

2. Discussion of Approach

In this section we examine the issues that arise in enabling secure ad-hoc collaboration and present solutions to some of these problems.

2.1. Secure Group Communication

As mentioned earlier, collaborators working in a collaboration environment must be able to reliably communicate with each other and know the identities of the other collaborators. Many systems support secure communication within a group using unicast mechanisms. This is accomplished by using a server or overlay network that relays messages to the members of a group. Another, more efficient method is to use reliable multicast.

Unicast vs. multicast. In this section we examine the advantages and disadvantages of solutions based on unicast and reliable multicast mechanisms.

Unicast mechanisms have the advantage that the existing network infrastructure supports unicast communication in a ubiquitous manner. In addition, developers are familiar with building applications and services based on unicast. Unfortunately, in the group setting, systems solely built upon unicast mechanisms suffer from efficiency, manageability, scalability, and/or fault-tolerance issues. Since messages are relayed within the group using pure middleware or application-level solutions latency is increased. Additionally, group members may be called upon to contribute additional resources, such as processing cycles in order to relay messages within the group. Also, relaying messages between participants in this manner raises further security issues. [8]

Central server-based systems, such as the Access Grid venue system, while easily manageable suffer from having a single point of failure. Peer-to-peer overlay networks, such as are used in Groove [9] are on the flip side of this problem. They address fault-tolerance, but are a management nightmare. Hybrid unicast solutions that combine a server and the overlay network approach (imagine Napster [9] with security) are tempting as they appear to balance these trade-offs. Unfortunately, these hybrids tend to inherit the problems of both types of systems. Also, the combination of centralized and decentralized security methods can lead to significant management problems.

A solution using reliable multicast mechanisms can provide better efficiency, fault-tolerance, and scalability than those using unicast mechanisms, but it suffers from two major disadvantages. First and foremost is the lack of ubiquitous IP multicast infrastructure deployment. Second, reliable multicast is unfamiliar to developers.

Our proposed solution to secure communication in a group environment is through the combination of the InterGroup protocols and the Secure Group Layer (SGL).

The InterGroup protocols build upon IP multicast capabilities in the network to provide reliable multicast mechanisms. They also include some additional mechanisms to address the disadvantages of an IP multicast-based approach.

Our architecture (Figure 1) allows participants that are not IP multicast enabled to participate as equal peers in the communication group.

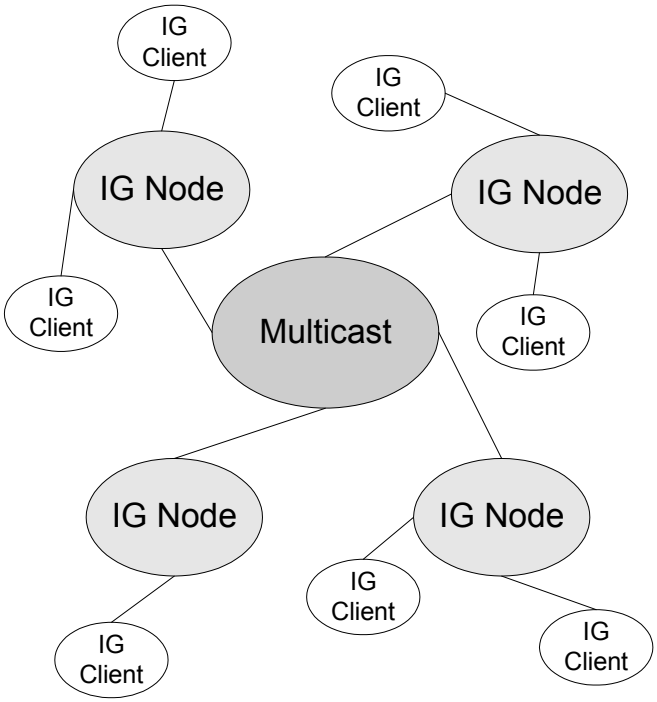

Figure 1. InterGroup architecture.

The InterGroup system consists of *nodes* and *clients*. The nodes run the core of the InterGroup protocols and communicate with each other using IP multicast. They also serve as entry points into the system for clients. The clients are light-weight processes that run on the participant's machine and provide the APIs for communication within InterGroup. Note that even though this architecture is hybrid, to the user, application developer, and security layer it appears to be a pure multicast.

The goal in designing the InterGroup protocols has been to provide membership services, reliable message delivery, and ordered message delivery to the application. These services are typical of group communication systems such as [4], [5], and [10]. The InterGroup protocols are intended to provide these application services in a wide-area environment with a large number of participants, prone to large latencies and frequent faults, such as the Internet. To accomplish this, they introduce an unusual approach to handling group membership, and support a receiver-oriented selection of service to enhance scalability. The levels of the message delivery service range from unreliable unordered to reliable timestamp ordered.

InterGroup is designed to operate in an asynchronous environment where no bound can be placed on the time required for a computation or for communication of a message. Processes may fail by stopping and taking no further actions. The network is allowed to partition and re-merge, and messages may be duplicated or reordered by the network.

At this time, the node software is implemented in Java, and clients in C++, Java and Python (as a wrapper to C++) are available.

Security. The Secure Group Layer (SGL) provides the collaborative application with a security context within which messages multicast over the wire can be cryptographically protected. The essential building block for setting up a secure multicast context is a distributed key exchange protocol that allows the participants to exchange a session key as equals and, therefore, treats them as peers. The first step in solving this problem was to design an algorithm that allows a set of participants to agree on a session key. [7] We refer to this kind of group genesis as the initial group Diffie-Hellman key exchange. Alone the group Diffie-Hellman key exchange is of relatively little practical use. A mechanism to enforce restrictions on who can participate in the key exchange and, therefore, the multicast group is needed. SGL integrates the Diffie-Hellman key exchange and access control mechanisms into a security layer.

SGL secures InterGroup in much the same way that the Secure Socket Layer Protocol (SSL/TLS) secures TCP. The approach is to interpose a security layer between the application and the transport layer protocol. This protocol is easy to deploy since it only requires minor changes in the application to convey the users' identity information and access privileges. SGL leverages off the properties of the underlying transport in transmitting its own messages. Because of this, SGL uses the same APIs as InterGroup, with the addition of some initial setting up of security parameters. These APIs have been developed for ease of use. The number of classes and methods has been kept to a minimum and the APIs mirror those of modern object-oriented datagram sockets as much as possible.

At this time, implementation of SGL is underway.

2.2. Authentication and Authorization

The Secure Group Layer provides the basic mechanism required to secure the group communication channel but it is only part of the final solution required. In an ad hoc environment, authentication and authorization are difficult problems. Most of the authentication and authorization systems available today depend on a central database of credentials and authorization policies. In an ad hoc environment this centralized database is not necessarily available.

The simplest means of dealing with ad hoc environments is for the group to use a shared secret to establish a session but this will not work in many situations and does not scale well. Another approach would be to have each member of the collaboration maintain their own authentication and authorization list containing the other members of the collaboration. This, however, is difficult to maintain and it is very easy to end up with a situation where the lists are inconsistent and none of the collaboration members can join the collaboration.

A likely solution will be the combination of a centralized database, a local version of the central database, and real-time authorization interfaces. With this combination, the local database might be consulted first, then central database next, and then the user would be directly queried if the first two methods failed. This combination of mechanisms would allow the user to incrementally build up the local database by storing each decision in the database for future use in authorization decisions. We plan to use our collaborative tools to investigate effective authentication and authorization structures for the ad hoc collaboration environment.

3. Case Studies

We now look at how our proposed solutions for enabling secure ad-hoc collaboration are being adopted in two projects. The first is *scishare*, an information sharing tool designed for dynamic groups of collaborators. The second is the Pervasive Collaborative Computing Environment (PCCE).

3.1. Scishare

In this section, we present an overview of *scishare* and the approach taken to enable secure ad-hoc collaboration within the context of this tool.

Groups collaborating on scientific experiments have a need to share information and data. This information and data is often represented in the form of files and database entries. In a typical scientific collaboration, there are many different locations where data would naturally be stored. This makes it difficult for collaborators to find and access the information they need. The goal of *scishare* is to create a lightweight information-sharing system that makes it easy for collaborators to find and use the data they need. This system must be easy-to-use, easy-to-administer, and secure.

The *scishare* information sharing tool is in many ways similar to existing peer-to-peer file sharing systems, such as Gnutella [9], Kazaa [14], Limewire [15], etc. Each peer designates a set of items to share within the system. Peers are able to search for items by sending a query to the network. The network delivers this query to the other peers, which run the query against the items they have designated to share. Metadata about the matching items is sent back to the peer that originated the query. This metadata contains information about the items, as well as how the actual items may be retrieved. Then the peer may go ahead and retrieve the item.

As part of our design, we allow for easy configuration of the message exchange patterns of the system. Currently, we use the InterGroup protocols to reliably deliver each query to all of the current participants in a scalable manner, without having to discover all of their identities. The transfer of the metadata and items is done using HTTP. Just as easily, we could transfer the metadata using the InterGroup protocols. We plan to make use of this configurability to perform measurements on the performance and resource consumption of different message exchange patterns.

In our design we also separated the application-specific requirements of information sharing from the more generic discovery mechanisms. We hope to advance and reuse the discovery component in other projects.

The prototype implementation of *scishare* is a Java application with a graphical front-end. It provides the user the ability to search for remote files (Figure 2), to transfer those files to the local machine, to manage the locally shared files, and the ability to save and restore the state of the application.

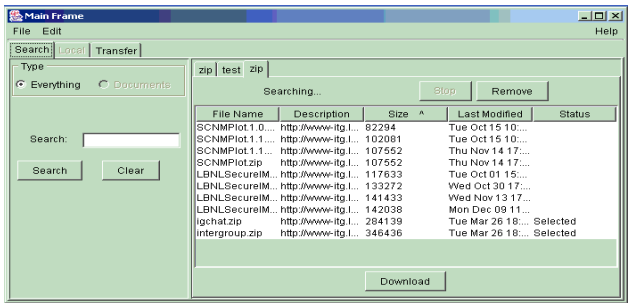

Figure 2. Performing a search in the *scishare* application.

The initial prototype of *scishare* sends queries and receives responses from a group of remote machines. The metadata in the responses describes the files, as well as the service required to retrieve them. The application uses the information in the response to transfer the file to the local machine.

We are currently in the process of incrementally adding security to *scishare*. We first use the concept of interposing a security layer between the application and the transport layer protocol in order to secure the communication. This means we will be using HTTPS instead of HTTP an SGL instead of InterGroup. Our design allows us to do this with little impact on the existing application code. Once complete, this will provide confidentiality, integrity, authenticity, and implicit authorization enforcement for the peer communication.

The next step is to tackle the issue of access control to the other resources, e.g. files. Our system will allow for individual peers to use the authorization services of their choice. The implementation that we provide will make use of Akenti distributed authorization [11] for this service. Akenti targets widely distributed environments where resource owners and users could span many autonomous organizations. It provides a powerful authorization policy language the enables fine-grained access control to resource in dynamic collaborations such as ours.

3.2. PCCE Secure Messaging

In this section, we present an overview of the PCCESecureMessaging tool and the approach we are

taking to enable secure ad-hoc collaboration within the context of this tool.

Collaborators have a need for a non obtrusive, lightweight, secure, and easy to use means of staying in contact with each other. Messaging applications have been found to be an effective tool for providing this collaboration contact. To this end, the PCCE project is developing a secure messaging application to support synchronous and asynchronous messaging. With our application users can hold group or one-to-one conversations on an on-going or ad hoc basis. These conversations may be public and open to anyone who is on-line or they may be private and open only by invitation. Although several messaging applications already existed, they do not incorporate asynchronous mechanisms and security. Our approach was to search for the open source messaging application as close to our needs as possible and modify it to add the necessary features.

The current implementation PCCESecureMessaging application 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 [16]) to replace its non secure TCP sockets with SSL connections. To provide persistence (e.g., unique nicknames and permanent channels) and enhanced presence information independent of any one environment, we developed a custom PCCE server which also provides authentication and authorization services. Both the IRC and PCCE servers use SSL network connections and X.509 credentials which they present to each other and to clients.

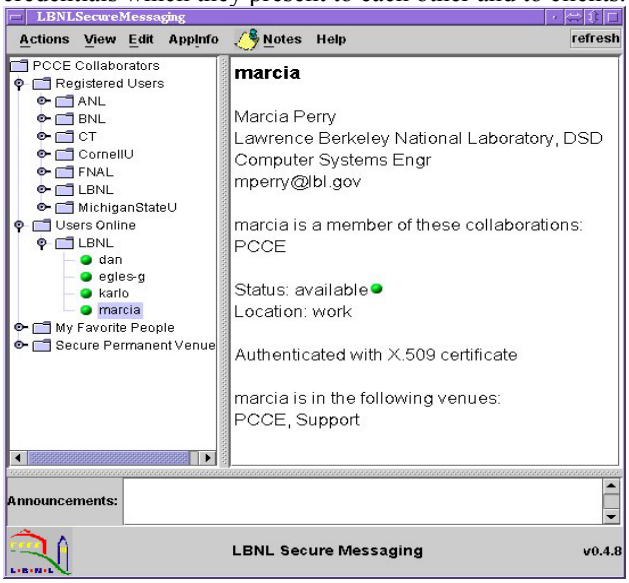

Figure 3. PCCE Secure Messaging Client

A view of the application is shown in Figure 3.

To access the servers, users must pre-register with the PCCE server with their username and password and certificate if they have one. Subsequent logins can be by either certificate or username and password. The PCCE server stores registration information in a local database and uses the information to make authorization decisions (e.g., who can connect to the IRC server, who can leave and receive notes, and who can perform administrative operations).

This current design forces users to rely on the PCCE and IRCD server to operate. This architecture presents a single point of failure for the system and impedes scalability. It also makes it difficult to quickly use the tool to support ad hoc collaborations.

We plan to convert the PCCESecureMessaging application to use InterGroup and SGL for messages on the channels and persistence information. This will allow it to operate without the PCCE and IRCD servers. By removing the dependence on these servers, PCCESecureMessaging will gain the ability to run in a purely ad hoc manner. The PCCE server, if present, is still valuable since it will enhance the functionality of the collaboration by providing persistence. The IRC server can be removed completely or used as an alternate means of connecting in participants that are not multicast capable. As we change to the group communication mechanisms we will also begin to investigate models for doing distributed authentication and authorization.

We are currently in the process of enhancing the PCCESecureMessaging application and extending the security architecture to allow incremental trust of users [2]. When we begin to integrate the InterGroup and SGL capabilities, the added dimension the group communication mechanisms will bring to the model will need to be considered.

4. Conclusion

By using InterGroup and SGL as core communication services in the collaboration environment, existing collaborations can easily operate in either an ad hoc or infrastructure-enabled setting without sacrificing security. This allows servers to provide added value services rather than being essential components. Thus, the dependence on centralized infrastructure is reduced and informal, spontaneous collaborations are enabled.

We have shown examples of two collaboration tools using these communication services. One, *scishare*, was designed with these services in mind. The other, PCCE, started out using a client-server model and is now adapting to using InterGroup and SGL as core communication services.

5. Acknowledgments

This work was supported by the Director, Office of Science, Office of Advanced Computing Research, Mathematical Information and Computing Sciences Division, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. This document is report LBNL Report number LBNL-52895.

6. References

[1] D.A. Agarwal, O. Chevassut, M.R. Thompson, G. Tsudik, "An Integrated Solution for Secure Group Communication in Wide-Area Networks", *Proceedings of the 6th IEEE Symposium on Computers and Communications*, Hammamet, Tunisia, July 3-5, 2001, pp 22-28.

[2] D. Agarwal, M. Lorch, M. Thompson, M. Perry, "A New Security Model for Collaborative Environments", *Proceedings of the Workshop on Advanced Collaborative Environments*, Seattle, WA, June 2003.

[3] D. Agarwal, C. McParland, M. Perry, "Supporting Collaborative Computing and Interaction", *Proceedings of the Grace Hopper Celebration of Women in Computing 2002 Conference*, Vancouver, Canada, October 9-12, 2002.

[4] D.A. Agarwal, L.E. Moser, P.M. Melliar-Smith, R.K. Budhia, "The Totem Multiple-Ring Ordering and Topology Maintenance Protocol", *ACM Transactions on Computer Systems, Vol. 16(2)*, May 1998, pp. 93-132.

[5] Y. Amir, C. Danilov, J. Stanton, "A Low Latency, Loss Tolerant Architecture and Protocol for Wide Area Group Communication", *Proceeding of International Conference on Dependable Systems and Networks (FTCS-30, DCCA-8)*, New York, NY, June 25-28 2000.

[6] K.Berket, D.A. Agarwal, O. Chevassut, "A Practical Approach to the InterGroup Protocols", *Future Generation Computer Systems, Vol. 18 (5)*, Elsevier Science B.V., 2002, pp. 709-719.

[7] O. Chevassut, "Authenticated Group Diffie-Hellman Key Exchange: Theory and Practice", PhD Dissertation, Universite Catholique de Louvain, Belgium, October 2002.

[8] M. O'Neill, et al., *Web Services Security*, McGraw-Hill/Osborne, Berkeley, CA, 2003.

[9] A. Oram, editor, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly & Associates, Inc., Sebastopol, CA, 2001.

[10] R. van Renesse, K.P. Birman, S. Maffeis, "A High Performance Totally Ordered Protocol", *Proceedings of the International Workshop on Theory and Practice in Distributed Systems*, Springer-Verlag, Dagstuhl Castle, Germany, September 1994, pp. 33-57.

[11] M. Thompson, A. Essiari, S. Mudumbai, "Certificate-based Authorization Policy in a PKI Environment", *to appear in ACM Transactions on Information and System Security*.

[12] "Access Grid", http://www.accessgrid.org/.

[13] "A Scalable and Secure Peer-to-Peer Information Sharing Tool", http://www-itg.lbl.gov/P2P/file-share/.

[14] "Kazaa Media Desktop", http://www.kazaa.com.

[15] "Limewire: The Most Sophisticated File-Sharing Application", http://www.limewire.com.

[16] ftp://ftp.blackened.com/pub/irc/ircservers /hybrid/old/