An Architecture for Conference-Support using Secured Multicast

Thomas Hardjono, Naganand Doraswamy and Brad Cain
Bay Architecture Laboratory, Bay Networks
3 Federal Street, Billerica, MA 01821, USA
Tel: +1-978-916-4538, Fax: +1-978-916-0620,
{thardjono,naganand,bcain}@baynetworks.com

Abstract

The current work argues that from a security perspective there is much to be gained by employing a "secured" IP multicast at the Network layer to support the formation and management of secure conferences at the Application layer. A secured IP multicast -- with group authentication and confidentiality -- already achieves a reasonable level of security, and therefore fulfils a large part of the basic requirements of secure conferencing. If host-to-host authentication and confidentiality has been achieved through an N-to-N multicast that has been secured, then to a large extent the basic security needs of conferencing has been satisfied. What remains would be for the other conference-specific security requirements to be satisfied using methods which are particular to a given conference scheme, such as cheater detection/identification methods based on cryptographic techniques. In the current work we propose an architecture called the Multicast/Conference Security Architecture (MCSA) to facilitate the use of (a secured) IP multicast at the Network layer for establishing (a secured) conference at the Application layer.

Keywords

Secure multicast, secure conferences, cryptography, routing protocols, key management

1 INTRODUCTION

The issue of group-oriented security has been a topic of interest in the field of computer and network security for the last two decades. This has been facilitated

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35388-3_42

H. R. van As (ed.), *High Performance Networking*

by a number of factors, including the growth of the Internet, the development of cryptosystems (in particular, public key cryptosystems), and the development of hardware and software for desktop level computing. Increasingly users are using the Internet not only for exchanging messages, but also for more complex interactions and as a forum for decision-making.

In this paper we attempt to bring together the main areas of research and development related to group-oriented security. The first area consists of solutions and schemes which are known collectively under *group-oriented cryptography*. These cover, among others, conference key distribution systems (eg. Ingemarsson, Tang & Wong 1982, Koyama & Ohta 1987, Steiner, Tsudik & Waidner 1996), digital multisignature schemes and secret sharing schemes (Simmons 1992). Most of the conference key distribution schemes that have been proposed require extensive cryptographic operations and have been designed with the Application layer in mind. These employ user authentication techniques either separately or integrated into the conference key distribution scheme. Some rely on the use of smartcards (eg. Koyama et al. 1987) as a user authentication technique and as a medium to store the conference security parameters.

The second area of development closely related to group-oriented communications is multicast, more specifically IP multicast. Here, group security is to be achieved through the distribution and management of cryptographic keys at the Network layer, using approaches which we broadly term *group key management* (GKM) protocols (Mittra 1997, Harney & Muckenhirn 1997, Ballardie 1996, Harkins & Doraswamy 1997).

In this paper we argue that from a security perspective there is much to be gained by employing a "secured" IP multicast at the Network layer to support the formation and management of secure conferences at the Application layer. A secured IP multicast -- with group authentication and confidentiality -- already achieves a reasonable level of security, and therefore fulfils a large part of the basic requirements of secure conferencing (see Section 2.1). If host-to-host authentication and confidentiality has been achieved through an N-to-N multicast that has been secured, then to a large extent the basic security needs of conferencing has been satisfied. What remains would be for the other conference-specific security requirements to be satisfied using methods which are particular to a given conference scheme, such as cheater detection/identification methods based on cryptographic techniques. Other benefits that would emerge include the increased efficiency of the conference formation and the possibility of the simplification of the conference key distribution schemes being employed.

In the current work we propose an architecture to facilitate the use of (a secured) IP multicast at the Network layer for establishing (a secured) conference at the Application layer. The purpose of the architecture is to introduce components at the Session layer that will coordinate the creation of multicast sessions, the

obtaining of group-keys for the groups, and the "mapping" of the conference instance to the multicast instance, and other tasks. The architecture must also allow different secure conference key distribution schemes to be used, independent of the multicast protocol and group key management protocol at the Network layer.

In the remainder of the paper, we define a "conference" as the N-to-N communications occurring at the Application layer (or originated from events at the Application layer). Similarly, we define "multicast" or "IP multicast" as the N-to-N communications occurring at the Network layer. We use the term multicast "session" to include all multicast "groups" related to a session. Thus, for example, a session may have an audio group and a video group (as in the Mbone), and both would be considered together when dealing with security. Consistent with this convention, we thus distinguish between a "conference key" for a conference and a "group key" for a multicast instance. For simplicity of the discussions, in either of the two cases we assume that the key is a private key (ie. symmetric cryptosystems) and we assume that the key is used to encipher traffic among the parties involved. Other variations on the type of key can certainly be employed depending on the needs of the circumstances.

In Section 2 some background is provided, looking from the perspectives of both the Application layer and the Network layer. This is followed by the proposed architecture in Section 3. Section 4 closes the paper with some remarks and conclusions.

2 PERSPECTIVES ON CONFERENCES AND MULTICAST

Research and developments in the area of group-oriented security in the past two decades has largely focused on two major directions which can be viewed from the *Application-layer perspective* and from the *Network-layer perspective*, reflecting the two main locations in the communications software architecture where solutions have been suggested.

2.1 The Application Layer Perspective

Much of the research efforts carried-out on the Application layer revolved around the use of cryptographic techniques to achieve a fair and secure method to distribute cryptographic keys to participants of a conference. The key belonging to the conference is then to be used to secure (eg. encrypt/decrypt and/or authenticate) the messages exchanged between the conference participants. To these schemes we apply the broad term *Conference Key Generation and Distribution System* (CKGDS), which may or may not incorporate user-authentication. The conference membership is usually determined by the piece of secret information carried by (or assigned to) the users. Joining a conference can be signified by the user indirectly

applying the portion of his/her secret information towards the conference-key computation. (For example, the user may apply his/her secret key to a circulating irreversible "token" that accumulates the keys). CKGDSs are often coordinated by a *Conference Coordinator* (eg. the participant that initiated the conference or a trusted third party). Other approaches employ a trusted third party to fully generate and deliver the conference key.

There are a number of security requirements for secure conferences. Some requirements that are of interest to the current work include:

- *Source identity and source authentication*: a member of the conference must be able to verify the source (identity of the sender) and authenticate the message from the sender. This assumes that user authentication has been enforced.
- *Data confidentiality*: encryption of data for confidentiality must be available for all traffic should the conference members decide to use such means.
- *Participation non-repudiation*: depending on the nature of the conference, members of the conference must not be able to repudiate their presence in a particular conference. This is crucial when the conference arrives at a decision which is binding to all members who are present, and who should not, therefore, be able to deny this fact at some future time.
- *Sender/receiver non-repudiation*: both senders and receivers in a conference must not be able to repudiate the fact that they sent or received messages respectively.
- *Cheater detection and identification*: any members (or intruders) that cheat must be able to be detected and identified. There is a range of behaviours that can be considered as cheating. A typical example would be a member that wishes to participate in decision-making a conference, but who would like to avoid the commitments resolved in that conference, by way of not submitting the correct information (eg. security parameters) at the creation of the conference. Other examples include masquerading as other members of the conference.
- *Joining conferences securely*: new parties that are permitted to join a conference (eg. by policy) should be able to do so in a manner that does not compromise or reduce the security of the existing conference participants.
- Secure ejection of a member: when a conference wishes to eject a member, a secure method must be used, both to arrive at the consensus with regards to the ejection decision and to actually carry-out the ejection. The ejection of a member should not in any way effect the security of the remaining members.

Although outside the scope of the current work, another issue that is commonly related to conferencing is the *anonymity* of the participants of the conference. Should it be required, anonymity can be achieved at the conference level using the appropriate secure conferencing scheme where the identity of the actual user is hidden using a *pseudonym* (Chaum 1981), and where the identity and the pseudonym is linked through other pre-conference means (such as smartcards, for

example, in the scheme of Koyama et al. (1987)). Although in most circumstances the end-users of the conference service are humans who require the identity of the peers to be known, there are circumstances where pseudonyms can be used and be made legally binding should the conference require it.

2.2 The Network Layer Perspective

In contrast to the efforts based on extensive cryptographic operations to satisfy the security requirements of conferencing, the developments that strive to find a solution at the Network layer have originated from the need and desire to make IP-multicast itself secure, independent of the applications that may employ it. These have taken the form of *group key management* (GKM) protocols, and have largely focused on providing practical and implementable solutions based on the realization that the network has true physical limitations and that the user-response quality represents a major factor in the design. The GKM approach is driven by the realization that although some security mechanisms have been introduced into applications that make use of IP-multicast, in itself IP-multicast does not have any authentication and/or confidentiality features.

A number of GKM protocols have been proposed (eg. Mittra (1997), Harney et al. (1997), Ballardie (1996), Harkins et al. (1997)). Here, the idea is that the GKM protocol (running at each relevant host) distributes keys to all hosts of a multicast session. Each group is assigned a unique key, and all traffic within the group would then be encrypted using the group-key. Only group-authentication is thus afforded, not sender-authentication.

One of the fundamental aims of these GKM protocols -- which is an aim that we also subscribe to -- is to separate the group key management for multicast from the multicast service itself, thereby providing independence of the GKM from the multicast protocol that happens to be in use (eg. CBT of Ballardie, Francis & Crowcroft (1993), MOSPF of Moy (1994), and others). Although there have been a number of proposals for a group key management protocol, none at the moment are being used on a wide scale on the Internet.

Other developments towards securing communications at the Network layer have emerged in the form of IPSEC (Atkinson 1995) and its related security technologies such as the Internet Security Association and Key Management Protocol (ISAKMP) (Maughan & Schertler 1997) and the Internet Key Exchange (IKE) (Harkins & Carrel 1998). IPSEC and IKE are unicast technologies that aim to secure communications between a sender and a receiver at the Network layer. This is achieved through the creation of *security associations* (SA) -- linked to a *Security Parameters Index* (SPI) -- which identifies all the parameters (ie. encryption algorithm, algorithm mode, encryption keys, etc) required for the two parties to securely communicate. However, IPSEC is designed for unicast between

one sender and one receiver, and therefore is not fully satisfactory for the security needs of multicast. In particular, the current version of IPSEC does not provide for the creation of a single security association for an entire multicast group.

2.3 Classification of Multicasts and Conferences

In this section we broadly classify groups from the perspective of the Network layer and the Application layer, realizing in full that the various attributes of groups have different interpretations in different circumstances. It follows that there is no one solution that can solve the security needs of both multicast and conference. Hence our approach of an architecture which can support the various parameters of multicast and conferences.

2.3.1 Network Layer Perspective

Receiver view:

Open multicast

- Any host is free to join a group and to receive the group's traffic.
- Joining does not require explicit permission.
- The receiver host requests its local subnet router to join a given group (eg. using IGMP or similar group-membership protocol).

Closed multicast

- Explicit permission must be requested to join a group (eg. to the initiator of the group).
- Mechanisms must be employed to enforce permissions (for example, encryption of traffic within the group).

Sender view:

Open multicast

- Any host can initiate/create a new group[1].
- Any host can send to a group without being a member of the group[2].
- Any host can send to any group.

---

[1] Although a host can create a group, technically speaking the host does not have to be a sender or receiver in the group. For simplicity, we assume that a host that creates a group is at least a sender.

[2] This may or may not be desirable, as it may allow denial-of-service attacks to the group. However, the notion of an authorized non-member sending to a group may have its uses and benefits.

Closed multicast

- Not everyone can create a group, and it is desirable to have mechanisms to limit who can create/initiate new groups.
- Only members of a group can send to the group.
- Traffic may be enciphered as a way to enforce explicit permissions.

2.3.2 Application Layer Perspective

Receiver (Sender) view:

Open conference

- Any user can initiate a conference.
- Any user can join or receive (send) provided he or she reveals his/her verifiable identity to the conference Coordinator and other conference participants.
- Presence at the conference does not bind a participant to the outcomes of the conference.
- The user need only submit parameters that are sufficient for identification purposes.

Closed conference

- Any user can initiate a conference.
- By definition, only limited individuals can receive from (send to) the conference.
- An invitee must make explicit request to the conference Coordinator.
- The identity of an invitee must be verified before he or she can receive (send).
- The invitee must submit cryptographic parameters as part of the conference key generation and distribution scheme.
- Participation in the conference must be provable through the key generation and distribution scheme.
- A participant must not be able to repudiate his/her presence and participation in the conference.
- The conference outcome is legally binding.

3 MULTICAST SUPPORT FOR CONFERENCING

In the current work we propose an architecture to facilitate the use of (a secured) IP multicast at the Network layer for establishing (a secured) conference at the Application layer. The purpose of the architecture is to introduce components at the Session layer that will coordinate, among others, the creation of multicast

groups, the obtaining of group-keys for the groups, and the "mapping" of the conference instance to the multicast instance. The aim is also to provide protocol independence, in the sense that different secure conferencing schemes at the Application layer can be used independent of the multicast protocol and group key management protocol at the Network layer.

One important assumption in the proposed architecture is that the underlying multicast service model (and its associated routing structures) allows a host who is part of a group to also transmit messages to the group. This can be achieved by using a multicast protocol that allows a receiver-host to also transmit messages to the group. Thus, in effect the underlying multicast protocol performs the N-to-N multicast. Most IP multicast protocols today allow this to occur.

In the case that the multicast protocol is limited to providing unidirectional transmission, then an *overlay* of N instances of these 1-to-N multicasts must be created and managed. That is, N separate multicast "trees" (emanating from a source to the receivers) must be created. This overlay will result in the creation of a total of N^2 security associations. Due to its large resource consumption, we will not consider this second approach any further.

3.1 Multicast Conferencing Security Architecture

The current work approaches the issue of providing support for conferences using (a secured) multicast by introducing the *Multicast/Conferencing Security Architecture* (MCSA) that provide selectable components which reside at the Session layer. The MCSA components together act as an intermediary between the conference application program at the Application layer and the multicast-related protocols (host-side or client-side) -- such as the Group Management Protocol (GMP) and the Group-Key Management (GKM) protocol -- at the Network layer (Figure 1). Here we use the general term GMP to denote the group membership and management protocol that is being employed (for example, IGMP (Deering 1989)). The IPSEC and IKE protocols (Atkinson 1995, Harkins & Carrel 1998) are also used -- directly or through the GKM protocol -- for host-to-host key exchange and data confidentiality.

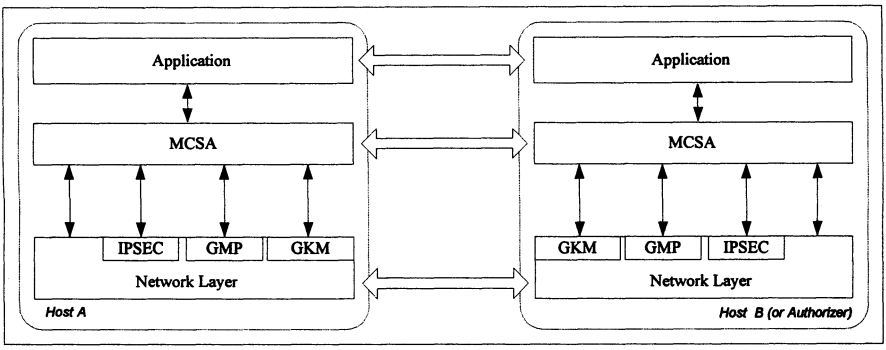

Figure 1: The Multicast/Conference Security Architecture

In practice the GKM protocol typically has a central point which authorizes and distributes keys for the group. Here we will broadly refer to that entity as the *Authorizer*, which may generally correspond to the *Group Key Controller* in the work of Harney et al. (1997), to the *primary core* in the CBT-based solution of Ballardie (1996), or to the *Key Distributor* (KD) in the scheme of Harkins et al. (1997). The Authorizer entity can be a router, a server or other devices.

In order to participate in the MCSA, the Authorizer entity must contain the components of the MCSA so that peer-level interaction would be possible. The Authorizer is assumed to run the GKM protocol to establish group-keys and perform high-level tasks, including certificate management, user access control, policy implementation, and others. The notion of an Authorizer is beneficial also from the point of view of security and access control policies, since the Authorizer can embody these policies. The Authorizer within a subnet can in fact be a subset of the *policy hierarchy* governing the entire autonomous system.

From the Network layer's perspective the Authorizer generates group-keys which are used by the group members to encrypt the payload within the multicast packets of a given group. Each group is assumed to have a secret (symmetric) key which is obtained by each member-host through a secure association (SA) with the Authorizer. Thus, further implied is the fact that each member-host must have a *distinguished name* (DN) (Adams and Farrell 1998) and must have a certificate before any security association can be established (Atkinson 1995). Hence, in our architecture we assume that each host has already been assigned a distinguished name and a certificate by the certification authority (CA).

3.3 Components of the Architecture

The MCSA consists of a number of components (Figure 2) which orchestrate the interaction between the multicast and the conference events. All parties involved in

the conference and multicast is assumed to employ the MCSA. Furthermore, we assume that the Authorizer contains all the peer elements at the Application layer and at the Network layer. The arrows in the diagram are simplified to denote interaction, both in terms of control/invocation and data/control flow.

The Conference Session Manager (CSM) and the Multicast Session Manager (MSM) are the two components that look after the relevant events occurring at the Application layer level and at the Network layer respectively (Figure 2). The CSM works in conjunction with the conference application and the conference key generation and distribution system (CKGDS). The CSM maintains the state information for each conference of which the user is a participant, and it maintains a database of the corresponding conference keys. Although the CSM does not actually use the conference keys, it has access to the database in order to maintain a correspondence between the instance of the conference (since the user maybe involved in several simultaneously) and the key for that conference.

The Multicast Session Manager (MSM) cooperates with the Conference Session Manager (CSM) in maintaining a correspondence between a conference instance and its underlying multicast instance. Depending on whether a host is the initiator of the multicast group or a receiver/sender in the group, the MSM has a number of tasks related to the creation of the multicast and the securing of it. The MSM is responsible for the initiation of the multicast group as a response to requests from the conference application. Through the session directory (SD) it coordinates the announcement of the new multicast group. It communicates with the peer MSM at the Authorizer in order to notify the Authorizer of the new group and request a new group key.

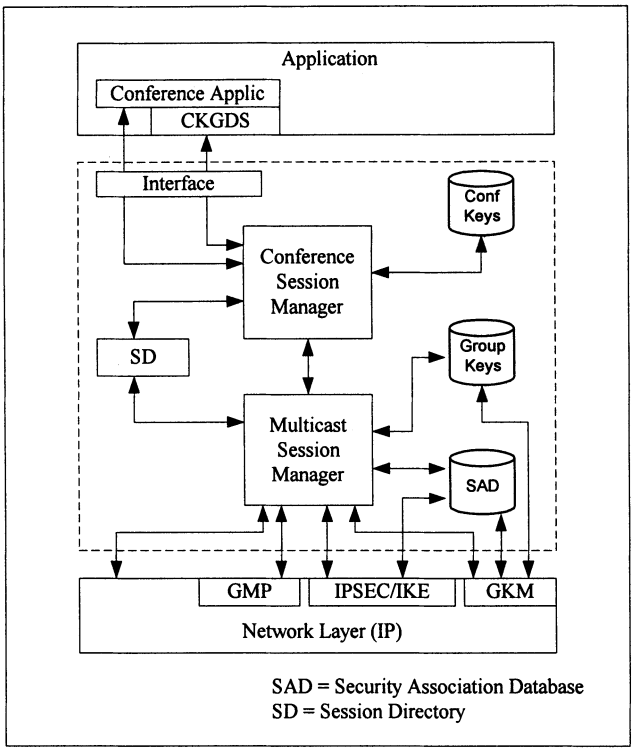

Figure 2: Components of the MCSA

A user that wishes to join a conference must instruct its host (implementing the MCSA) to first join the multicast group corresponding to that conference. Joining the multicast group can be achieved through the GMP. After the host becomes a group member, it must open a secure association with the Authorizer in order to obtain the corresponding group key. The MSM must maintain a mapping between the multicast instances (of which the host is a member) and the security association contained in the Security Association Database (SAD), the SAD being part of the IPSEC definition. Through the security association, a copy of the group key is obtained from the Authorizer and it is stored in the group key database. This group key is used to encipher (decipher) traffic at the Network layer destined for (received from) the multicast group.

Although not clearly shown, ideally the conference application and the corresponding CKGDS should deal with the MCSA through a suitable interface or API. Such an API should be usable with a variety of CKGDSs and may even be extendable to other applications that exhibit conference-like behaviours and security requirements.

3.4 Initiator and Sender/Receiver Interactions

In the current architecture, we assume that at the conference level, the user/application initiating the conference is the Coordinator of the conference and the contact-point for other users wishing to join or leave the conference. The issue of the ejection of conference members is also the responsibility of the Coordinator.

In the following we consider the MCSA as a unit from the perspective of a sender/receiver and an initiator of a conference. We assume that the initiator host is also where the conference Coordinator resides.

Initiator MCSA:

- Upon a conference-formation request from the application/user, the MCSA invokes the session directory (SD/SDP) to announce a new multicast session. The announcement must contain the identity of the Authorizer from which other hosts can obtain the group key. The announcement must also indicate that the multicast session will be part of a conference soon to be created. The application/user may also provide a list of acceptable (non-acceptable) users for the conference and the associated access control information.

- The MCSA creates a security association with the Authorizer and notifies the Authorizer about the new multicast session. The MCSA also provides it with access control information (for access at the user level and the host level). The Authorizer prepares a group key for the multicast session.

- The MCSA uses the security association to obtain a copy of the group key.

- The MCSA invokes IPSEC to encipher all subsequent traffic destined for the multicast session.

- After a given period of time (to allow other hosts to become members of the multicast session) the MCSA initiates the transmission of a conference-call message through the multicast session, with the announcement details being encrypted using the group-key. It names its own CKGDS as the Coordinator of the conference.

- The MCSA waits for the conference-call-responses from the members of the multicast session, and notifies its conference application and CKGDS about the group-members that request to join the conference.

- The MCSA notifies the CKGDS that all peer MCSA are waiting for the conference key generation and distribution to commence. The Coordinator CKGDS (at the Initiator host) then begins the conference key generation and key distribution phase, culminating in each application having a copy of the

conference key. Note that all traffic during (and after) the conference key generation/distribution are encrypted at the Network layer using the multicast group key.

Receiver/Sender MCSA:

- Upon user request the MCSA invokes the session directory tool to determine active multicast sessions. (ie. start time, duration, IP addresses, formats, etc). Included here is the address of the Authorizer. As an aside, the user may also request the Authorizer to provide it with a copy of the Authorizer's certificate (signed by an acceptable global authority) to prevent masquerading.

- The MCSA notifies the user/application about the active multicast sessions and asks the user to select the multicast sessions to join.

- The MCSA invokes the GMP to notify the local subnet router about the request to join a particular multicast session(s) which comprise a conference. (This step depends to a large extent on the multicast protocol being employed).

- Upon its host becoming a member of the multicast session, the MCSA invokes the GKM to obtain the group key. One way to obtain the key is to create a security association (SA) with the Authorizer, and then to use the resulting secured channel to download a copy of the key.

- The MCSA invokes IPSEC to decipher (encipher) all subsequent traffic received from (destined for) the multicast session using the group key.

- Upon seeing a conference-call message (issued through the peer MCSA at the Coordinator host) the MCSA responds to the call and notifies its own CKGDS of the ready-status of the Coordinator CKGDS. After some conference parameter negotiations, the CKGDS participates in the conference key generation and distribution scheme.

Note that in the current architecture the conference application can still employ its own mechanisms for confidentiality and authentication at the Application layer. This approach may be preferable in certain circumstances, where the security requirements are more stringent and where user-to-user security must be established (eg. using smartcards).

In the current architecture, the group key at the host level only affords *group authentication*. That is, other member-hosts can be assured implicitly only that the data originated from a valid group-member (unless other additional means is employed). If *sender authentication* at the Network layer is also required in order

to identify the source-host, then each member-host must embed its signature as part of the payload. This is because IPSEC does not provide explicit means to include signatures for authentication for each data packet. Digital signatures can be employed, while less explicit signing mechanism are also available.

4 REMARKS AND CONCLUSION

In this paper we have argued that from a security perspective there is much to be gained by employing a "secured" IP multicast at the Network layer to support the formation and management of secure conferences at the Application layer. A secured IP multicast -- with group authentication and confidentiality -- already achieves a reasonable level of security, and therefore fulfils a large part of the basic requirements of secure conferencing.

The current work has viewed and discussed group-oriented security from two perspectives, namely secure conferencing at the Application layer (via conference key generation and distribution systems) and group key management protocols at the Network layer. It has also attempted to classify multicasts and conferences in order to find similarities and differences in behaviour at the two layers.

Following from this the Multicast/Conferencing Security Architecture (MCSA) was proposed that provided a way to identify components at the Session layer that could act collectively as an intermediary between the conference application program at the Application layer and the multicast-related protocols (host-side or client-side) at the Network layer. Part of the internal task of the MCSA is to map instances of conferences to that of multicasts in a secure fashion with the aid of an Authorizer entity. The interaction of the relevant parties in the multicast and conference has also been briefly outlined.

There are still a number of open problems in the area of group-oriented security, particularly with respect to practical GKM protocols. These include the introduction of a *group security association* (GSA) for IP multicast in the spirit of IPSEC, the design of a GKM protocol for inter-domain key distribution suitable for the various IP-multicast applications, and others. These, as well as other issues specific to the MCSA, will be the directions for future work.

5 ACKNOWLEDGMENTS

We thank Jim Luciani for the important input and support for the current work. We also thank the anonymous referees for their extensive comments and insights into the issues discussed in the paper.

REFERENCES

Adams, C., and Farrell, S. (1998) Internet X.509 public key infrastructure certificate management protocols, March 1998. draft-ietf-pkix-ipki3cmp-07.txt available at `http:www.ietf.org`.

Atkinson, R. (1995) Security architecture for the internet protocol. RFC 1825, IETF, August 1995.

Ballardie, T. (1996) Scalable multicast key distribution. RFC 1949, IETF, 1996.

Ballardie, T., Francis, P., and Crowcroft, J. (1993) Core based trees: An architecture for scalable inter-domain multicast routing. In Proceedings of ACM SIGCOMM'93 (San Francisco, 1993), ACM.

Chaum, D. (1981) Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* 24, 2 (1981), 84--88.

Deering, S. (1989) Host extensions for IP multicasting. RFC 1112, IETF, 1989.

Harkins, D., and Carrel, D. (1998) The internet key exchange (IKE), March 1998. draft-ietf-ipsec-isakmp-oakley-07.txt available at `http:www.ietf.org`.

Harkins, D., and Doraswamy, N. (1997) A secure scalable multicast key management protocol, November 1997. draft-ietf-ipsecond-00.txt.

Harney, H., and Muckenhirn, C. (1997) Group key management protocol (GKMP) specification. RFC 2093, IETF, July 1997.

Ingemarsson, I., Tang, D. T., and Wong, C. K. (1982) A conference key distribution system. *IEEE Transactions on Information Theory* IT-28, 5 (1982), 714--720.

Koyama, K., and Ohta, K. (1987) Identity-based conference key distribution systems. In Advances in Cryptology - CRYPTO'87 (Lecture Notes in Computer Science No. 293) (1987), Springer-Verlag, pp. 175--184.

Maughan, D., and Schertler, M. (1997) Internet security association and key management protocol (ISAKMP), July 1997. draft-ietf-ipsec-isakmp-08.txt available at `http:www.ietf.org`.

Mittra, S. (1997) The Iolus framework for scalable secure multicasting. In Proceedings of ACM SIGCOMM'97 (1997), ACM, pp. 277--288.

Moy, J. (1994) Multicast extensions to OSPF. RFC 1584, IETF, 1994.

Simmons, G. J. (1992) An introduction to shared secret andor shared control schemes and their application. In *Contemporary Cryptology: The Science of Information Integrity*, G. J. Simmons, Ed. IEEE Press, 1992, pp. 441--497.

Steiner, M., Tsudik, G., and Waidner, M. (1996) Diffie-Hellman key distribution extended to group communications. In Proceedings of the 3rd ACM Conference on Computer and Communications Security (New Delhi, March 1996), ACM.