Elements of Trusted Multicasting

Li Gong and Nachum Shacham

SRI Computer Science Laboratory
Menlo Park, California 94025, U.S.A.
{gong,shacham}@csl.sri.com

Abstract

Multicast is rapidly becoming an important mode of communication as well as a good platform for building group-oriented services. However, to be used for trusted communication, current multicast schemes must be supplemented by mechanisms for protecting traffic, controlling participation, and restricting access of unauthorized users to the data exchanged by the participants. In this paper, we consider fundamental security issues in building a trusted multicast facility. We discuss techniques for group-based data encryption, authentication of participants, and preventing unauthorized transmissions and receptions.

1 Introduction

Emerging distributed applications, such as multimedia teleconferencing, computer-supported collaborative work, and remote consultation and diagnosis systems for medical applications, depend on efficient information exchange among multiple participants. Network-based multi-destination switching is an essential mode of communication for such applications.

Little concrete consideration has been given to the security and privacy issues brought about by multicasting, resulting in sessions without any control over who participates and the level of participation. Traditionally, teleconferencing has been conducted over private networks or dial-up lines, which provided some physical security measures. However, in integrated networks like Internet or ATM, multicast traffic shares network resources with other traffic. Such integration allows a quick setup and flexible maintenance of multicast sessions, but it also brings about serious security and trust issues.

In the current Internet-based IP-Multicast design [5, 2], for example, a broadcast message is not guaranteed to arrive at only the intended destinations. There is no security control as to who can register a (dynamic) session address, who can or cannot join a session or inject traffic into the session, or how to deal with the security implications when a host leaves a session. Furthermore, no security management is specified for the multicast hosts or routers. Several ad-hoc fixes has been recently proposed, but no systematic security architecture design is available.

Some of this openness was done by choice to facilitate large-scale distribution of unrestricted data, as the successful deployment of IP-Multicast demonstrates. However, many multicast sessions in military, business, commerce, and medicine are not feasible in such an open environment, and will require controlled participation and data privacy (possibly at multiple security levels).

What is needed is a trusted multicast architecture (with associated procedures) to handle these security issues in a coherent manner and to protect network services and applications. Careful investigation of such an architecture can save duplicated, and quite possibly unsophisticated, efforts of inserting security technology into individual applications.

In the rest of this paper, we consider fundamental security issues in building a trusted multicast facility. We discuss new security challenges in multicast and currently available solutions for them. We conclude with an overview of related prior work and directions for future work.

2 New Challenges in Multicast

Due to page limit, we skip an overview of existing network multicast mechanisms and refer the reader to the following references [5, 6, 15, 2].

In multicast, just as in unicast, an adversary (who can be a legitimate network user) may eavesdrop on confidential communication, disrupt or distort a session's data exchange, inject unauthorized or bogus traffic, block a session's progress, masquerades as someone else to join in a session, or initiate and operate a bogus session. Therefore, the privacy, integrity, availability, and authenticity of a multicast service must be protected.

However, the nature of multicast presents new problems that cannot be effectively dealt with using trivial extensions of techniques for secure unicast. For example, in setting up a secure point-to-point communication channel, one knows the identity of the party at the other end. In a multicast session, knowing who are present within the session is typically not guaranteed, let alone controlling session membership. Another problem is group-oriented secure data exchange. Although using $O(n^2)$ end-to-end secure channels can provide secure group communication, such an architecture loses all the advantages a multicast facility has over unicast. Also, in group-oriented communication, additional mechanism is needed to reliably establish the identity of the originator of a message. Moreover, group-oriented authentication (and key distribution) is not necessarily $O(n^2)$ pairwise authentication, because of the many possible interpretations of the meaning of belonging to a multicast session.

In short, new protocols are needed to perform security functions for controlling session organization and management, secure broadcast, and user access to the network. We will elaborate on these functions and protocols in the next section.

3 Trusted Multicast

The architecture design should consider the placement of the many control functions at appropriate network protocol layers and appropriate network sites, and the trust relationship between network sites and hosts (some of which may function as multicast management centers).

Apart from satisfying the security requirements mentioned earlier, a desirable design should also be *compatible* with existing network protocols, *scalable* to the scope of the global Internet, and *transparent* so that higher level applications and services are free from concerns for details (such as authentication, and the checking of credentials and policies). In particular, the security mechanisms residing at the session and presentation levels will place no restriction on the underlying networking mechanisms. As such they will coexist with networking standards and will not depend on modification of transport and switching elements in the (inter)network. Moreover, the architectural design should be *localizable* in that an area of a network can install the trusted multicast facility and achieve firewall-style self-protection even if other portions of the (inter)network do not yet support trusted multicast. This feature is important to facilitate a gradual introduction of the new technology to the existing millions of host machines. Finally, the architecture has to be *flexible* to support a variety of policies of session control as required by higher level applications. These features together will make trusted multicast easier to be integrated into an existing environment such as the Internet.

3.1 Authentication

Authentication – a process by which one satisfies another about one's claim of identity – is an essential mechanism in trusted multicast to control the registration of a multicast address and screen hosts that request to join an existing session. Often, the requesting party *and* the existing session members need to authenticate each other. Sometimes one party may delegate some of its authority to another party. This can be achieved by letting the former issue a credential, such as a signed and verifiable certificate, to the latter who later presents such credentials together with a proof of its identity.

There is a very rich body of literature on the topic of authentication and delegation in distributed systems (see [1] for references). Thus we only concentrate on the extension of the basic pairwise authentication model to the internal and external authentication of all existing members of a multicast session as a group.

There are two distinct issues here. One is who controls the membership of a group and how to enforce such a membership policy. The other is what constitutes being admitted to an existing group. We tackle the second issue first.

Except in the case of anonymous sessions, a minimum requirement for belonging to a group is that the new party should be introduced to (some or all) existing members. This introduction merely announces the identity (i.e., the name) of the new party, and can be done by any member, by at least some preset number of members, or by a group leader. Whoever handles the introduction process in essence implements the admission policy. A more stringent requirement can be that each existing member and the new member must successfully authenticate each other after an initial introduction. An even more strict requirement can be that every member must also know that every other member knows about the new party.

To require pairwise authentication, there need to be O(n^2) runs of an authentication protocol for a group of size n, although such runs are done incrementally as new members join the group. A more efficient method is to elect a group leader who is trusted (by other group members) to properly authenticate the new member. In this case, the group leader in effect controls the group membership policy. A natural choice of the leader is the session initiator. When it registers the session, it becomes the leader and specify an admission policy. If the leader crashes or retires, a new leader must be elected through a secure protocol, possibly according to the session policy. It is conceivable that some members may monitor the leader's actions for checking conformance with the policy.

When a member voluteeringly or otherwise leaves a session, especially long running sessions, the member should properly "sign off" and erase any sensitive information related to this session, such as the group key. Otherwise, this member can later eavesdrop without explicitly joining the group, or the residue information a departing member leaves behind may be exploited by potential attackers. Since a benign member may forget to clear up, it is always prudent to update the group key at this point. For the same reason, a member must be garbage collected when a proper sign-off fails to take place. In other words, the group may sign him off to make sure that he cannot participate in future discussions. The status of members can be monitored by a number of methods, such as by exchanging "heart-beat" messages.

3.2 Secure Sessions

Based on (our and others') previous experience in the area of secure group-oriented distributed systems, we can identify several crucial issues in forming secure sessions, which we now discuss in some detail.

Session membership policies. Sessions can be set up for various purposes with different membership policies. Below are a few "typical" policies for managing multicast. At one end of the spectrum is a totally open policy. Anyone is free to take part in such an open session, which resembles a mass gathering or an unmoderated USENET newsgroup. A more restricted session, as a presidential town hall meeting, has open participation but members have to pass a security check and there is a session moderator. An even more restricted session is like a board meeting where only board members can participate. From time to time non-members are invited, but only if a majority of the members consent. We can easily imagine an infinite number of such policies (including the multilevel security variety) and, at the other end of the spectrum, is an ultra secure and closed session. As we argued earlier, the architecture should be flexible enough to accommodate many policies so that each individual application or each session can choose to implement one policy or a set of basic policies.

In some applications, membership information itself is sensitive, thus rules have to be defined for disclosing the current group membership. However, traffic analysis may still reveal membership. One solution is to have trusted gateways that will mask all sources and destinations. Another solution is to saturate the network so that an observer cannot distinguish between a meaningful packet and a junk one.

Registration and deregistration. When a host or user requests a network address to establish a multicasting session, a multicasting management center (MMC) must verify the requester's identity and check the session policy dictating who can register a session. Similarly, the requester may also wish to verify the identity of the MMC so as to be sure that the session is correctly registered.

Information such as membership policy, user credentials, and records of registered sessions can be stored at the MMC in the form of access control list.

The MMC can be a centralized service or a distributed one. When there are more than one MMC in a network, they must coordinate among themselves to ensure consistency and uniqueness of multicast addresses. They may or may not adopt an identical admission policy.

When a request to deregister a session arrives at an MMC, it must authenticate the requester and determine if it is authorized for such a request. Normally, only the session leader, or its representative, can deregister the session. When the session leader crashes or when it forgets to deregister, a "dead session" must be garbage collected. This can be done by having the MMCs periodically checking the activities of all registered sessions. Similarly, if a multicast session is deemed undesirable (e.g., when it floods the network with junk), the session must be terminated.

Joining and leaving a session. The communications among session members can be public or private. This can be specified at registration time, and can be changed later. It is also possible to leave this attribute to the discretion of the individual members, so that some communications may be private while the rest remain public. For private sessions, adequate privacy provisions are required in multicasting. Because most networks are open in the sense that anyone may eavesdrop on network traffic we need to encrypt message contents, when requested, to ensure that only those authorized can correctly decrypt them.

Secure session communication. There are two strategies for this purpose. The traditional approach is to arrange the distribution of a common encryption key, shared by all session members, and encrypt broadcast messages with this group key.

With this approach, an initial key distribution must take place when a session is registered. This key can be chosen either by the session leader or the authentication server. The key is then maintained by the session leader or stored at the MMC where the session is registered. When someone applied to join a session, the group leader or the MMC checks the session policy to make a decision. When a new member is accepted into the session, it receives the group key from the MMC or the group leader. When the group leader crashes, a new leader is elected, using any of the many known election protocols.

A significant shortcoming of this method is that, if session members can be dishonest, the group key does not identify the source of a message. Another problem is that frequent key change (and their synchronization among group members) during a busy session is obviously not very desirable for performance reasons. Secure broadcast algorithms alleviate these two issues, as we describe below.

Secure and efficient broadcast. When a member can choose a new encryption key for a new membership on a per message basis, the risk of key leakage by a departing member is considerably reduced and a new group key is unnecessary.

Suppose each member has a pair of public and private keys, for example, for use in the RSA cryptosystem [13]. Each member must have knowledge of the group membership – at least he must know the public keys of all the other members. To broadcast a message, a member chooses an encryption key (for use in DES) at random and encrypts the message. The member then signs a timestamp (to defend against replay attacks), the DES key, and a one-way hash function [12] of the original message (to serve as an integrity checksum), using its private key. The member then encrypts this signature with each other member's public key. Finally, the member broadcast (or multicast) the $(n-1)$ encrypted signatures and the encrypted message. (Here, n is the size of the group.) At the receiving end, a group member decrypts a suitably encrypted signature (with its private key), obtains the DES key (using the broadcaster's public key), and then retrieves the message. Clearly, anyone who is not a group member would not be able to retrieve the message. This simple algorithm has the following attractive features: (1) the encryption key can be changed for every message, thus a membership change does not require additional security measures such as changing the group key; (2) only the message header (containing the encrypted signatures), and not the message body, increases linearly in length with the number of destinations. Moreover, the computation of a one-way hash function can be very efficient, and the checksum adds only a few extra bytes [12].

To use this broadcast primitive, session members receive and cache others' public keys and other relevant data. When notified of session membership changes, members simply update their local membership lists and choose new DES keys for future multicasts, which are thus made unreadable to those outside the session (including the members who just left).

If group members do not have public key capability, the above algorithm cannot be easily extended since it appears that not only every pair of members must share a distinct key, thus bringing about the $\mathrm{O}(n^2)$ key problem, but also the message body need to be encrypted $(n-1)$ times (e.g., using DES) and thus the broadcast message body also increases with the size of the group, making the algorithm less scalable.

Nevertheless, there exist secure broadcast algorithms (see [8] for references) with the same attractive features of the one using public-key cryptosystems. Due to page limit, we do not discuss these algorithms further except by noting that the message header in the NED-B(n) protocol [8] has $O(n \log n)$ bits. This algorithm is thus more space efficient compared with the use of public-key systems where the message header uses $O(c \times n)$ bits where c is the greater of $O(\log n)$ and the size of a public key signature (typically 512 bits or longer).

The NED-B(n) algorithm is also far more computationally efficient in that the complexity is merely $O(n(\log n)^2)$, whereas to use public key systems, n digital signatures must be signed and verified. Notice that each pair of members must now share a distinct encryption key, thus the $O(n^2)$ keys problem seems to persist. However, there are methods to reduce the complexity from $\mathrm{O}(n^2)$ to $\mathrm{O}(n)$ [10].

Encryption mode. It is generally undesirable to use the basic Electronic Code Book (ECB) mode because each block is independently encrypted thus the mode is more vulnerable to slicing attacks (e.g., by reordering blocks) and cryptanalysis.

In the plaintext-feedback mode, the encryption of a block depends on the plaintext of the preceding block(s). This results in infinite error propagation in that a block cannot be correctly decrypted unless all preceding blocks are correctly decrypted. In other words, a single lost packet (or cell) or any data corruption during transit renders the rest of a message unreadable. This undesirable property may necessitate excessive retransmission when the network or a stream is unreliable.

The ciphertext-feedback mode, on the other hand, has a limited and controllable error propagation, typically of 2 encryption blocks. As long as 2 consecutive ciphertext blocks are received correctly, decryption can proceed and resynchronization is automatic. Moreover, in some applications involving audio and video, occasional packet loss does not matter much because it causes only minor degradation in voice or video quality. All these made this encryption mode the favorable choice.

The multicast facility may need to adapt between multiple encryption modes, and between using block ciphers and using stream ciphers, according to the application environment. For example, ECB mode allows random access to individual blocks of a message. Also, a stream cipher, especially if precomputation of the stream is possible (e.g., using DES as a pseudo-random number generator), is faster than a typical block cipher, but due to the cost of resynchronization when data loss occurs, it is more suitable in a highly reliable environment.

Finally, we consider the case of partial encryption of streams. One motivation to avoid encrypting the whole stream is efficiency, e.g., when it is possible to identify crucial data (such as control information) that are the only portion to be encrypted for security. The sending and the receiving ends must be in synchrony as to which portion is encrypted and which is not.

Another motivation for partial encryption is that not all receiving ends require the same amount or type of information from the source. One member may want to hear the voice more clearly while another wants to improve the image quality.

Similarly, different portions of the same stream may need to be encrypted differently. For example, the background of a picture may be public knowledge, but the human image in the center may be highly classified and need special treatment.

In such cases, it is conceptually simpler to view partial streams as many layers of the same stream, and techniques developed for Heterogeneous Multicast (HMC) [14] can be used to explore the concept of multicast sessions in which users participate at different bandwidth levels.

Session advertising. Many services need to be advertised. The existence of a registered session can be announced, e.g., by a bulletin board service. Proper credentials and identification information must be provided so that one can verify the authenticity of the advertisement. Sometimes it is necessary to restrict the dissemination of service information. For example, the bulletin board can be made multilevel secure.

3.3 Control of Tree Access

Authentication guarantees that only certified users become legitimate participants, and group-oriented encryption enables only those participants to decode the data that is being sent. Yet, the current network level multicast routing and forwarding mechanisms allow any user to establish a path to the tree and to inject traffic at will. It is thus of interest to employ network level mechanism that limit physical access to the distribution tree.

A simple approach, used by ST-II [15], is to require all tree attachments to be initiated by the source of the tree. This makes the source aware of who is listening and allows it to reject unwanted requests. If the source approves a connection request, it sends a message down the tree toward the destination. At one point, the message leaves the tree and establishes an extension path while progressing toward the destination. While this technique provide some level of control, it adds significant processing load to the source thereby limiting the session size.

In a teleconference, receivers usually initiate attachment to a specific distribution tree. If the distribution is by multiple source-based trees, a typical receiver is likely to shift attention from source to source thereby changes frequently its attachment to the respective source-trees. Even if the distribution is over a single core-based tree [2], a receiver changes its resource allocation on that tree to reflect the shifting emphases on different sources. Both resource allocation and establishing and disconnecting path involve actions by multicast agent residing at network nodes.

As was argued above, requiring the sources to mediate these actions reduces efficiency and performance. It is thus desirable to allow receivers to interact directly with the multicast agents with minimal if any involvement by the sources. This receiver-agent interaction must be supported by proper authentication measures that prevent unauthorized users from directing traffic in their direction.

The process of authentication of end hosts to network switches is somewhat different from authentication among peer hosts. First, due to storage and processing limitations, switches cannot be expected to maintain keys, public or private for all hosts connected to the network. Moreover, at a time a connection is initiated, a host does not know which switches will actually modify their routing tables to support its request. That is, advance key exchange with the "right" switches is not feasible.

Each network switch could maintain a small number of public keys, with the corresponding private keys held at one or more authentication servers. Before initiating a request for a new connection or modification of an existing connection, a host first send the request message to the authentication server. After authenticating the host, the server sends back the message with a signature attached. That message is sent along the network path, and every switch on the way checks it against the signature, using its locally held public key of the server.

Similarly, a node that wishes to inject traffic must first obtain a signature from the server. A network node will not accept traffic from a source without such a signature.

Notice that authentication of receivers are done relatively infrequently and are not likely to impose a great processing burden on the switches. In contrast, to block unauthorized sources, packets must carry a signature, which results in a much heavier load on the switches, especially in the case of stream traffic, e.g., video. For stream traffic switches, only a fraction of the packet should carry signatures, and the switches should cache the signatures for forwarding of those packets that do not carry signatures. The cache should be refreshed every so often.

It is also conceivable that malicious or stupid parties may misuse network resources, for example, by flooding a multicast address, or by setting up a bogus multicast session with the sole purpose of using up network bandwidth. Simply ignoring such "junk" packets (like people throwing away junk mail) is inadequate because by the time such packets reach their destinations, precious network resources have already been consumed. Thus we need to block such usages (as soon as they are identified) as near to the multicast source as possible. We can imagine that the MMCs acting as warning posts that send out advisory messages to individual hosts, who in terms regulate the local network access it controls. Many Internet sites have already started filtering incoming and outgoing packets, but mostly for different reasons. Alternatively, ideas similar to those developed for the Visa protocol [7] can also be used. However, blocking misuse seems to require modifications to the existing multicast (and unicast) protocols and softwares.

3.4 Scalability and Trade-off

The task of assigning initial encryption keys for user or host authentication grows only linearly with the number of users and hosts interested in participating in secure multicast. The number of messages for establishing a session and for changing the session membership is also linear to the size of the session. A secure broadcast algorithm (e.g., NED-B(n)) allows dynamic encryption key change on a per-message basis, thus eliminating the need for a new group key upon group membership change. Software implementations of encryption algorithms are sufficiently fast while hardware implementations will further enhance performance. More importantly, in protocol NED-B(n), the most costly activity — the encrypting of broadcast messages — remains constant when the session size increases. Only the message header (which contains instructions for decryption) increases in length, and only linearly, when the session size increases.

Nevertheless, at present, a quality public-key cryptosystem such as RSA typically has a key length of 512 bits. This means that a multicast message header increases 64 bytes for each additional member. Such an increase may not be practical in some situations. One trade-off is to use short RSA keys or use different public key algorithms. Another is to selectively (and dynamically) choose between using a group key and using secure broadcast, depending on the importance of each session and perhaps each message. For example, when group membership becomes reasonably stable, it is more efficient to use a group key.

Sometimes, members of a session also forms a large number of (sub)groups. For example, in a teleconferencing application, although everyone's voice is broadcast to all destinations, a member may wish to broadcast his physical location information (his coordinates) only to members in his proximity. To structure such (sub)groups, the secure broadcast algorithm has the advantage that, since data encryption keys can be chosen dynamically on a per message basis, a member need not store a potentially large number of group keys. Of course a member can optimize by storing data encryption keys and the corresponding signed message headers and reusing them until a membership change occurs.

3.5 Multilevel Secure Multicast

End-to-end encryption can be used to control message dissemination on a discretionary basis. Mandatory control, possibly of a multilevel structure, can also be implemented. Here, groups and members are classified at different levels, and a party cannot become a member of a group whose security level is higher than its own. Levels are also assigned to encryption keys. A member cannot have access to a key whose level is higher. For encryption, the level of the key must be equal to or higher than that of the plaintext. The level of a ciphertext is then the level of the key. Such an arrangement effectively segregate the multicast network into virtual networks, each at a distinct security level.

Multilevel security may require that lower level information be accessible to higher level group members. One way to achieve this is for the lower level group members to be able to "write up" – to send multicast messages to groups at a higher level. However, since the sender typically cannot have the suitable higher level encryption key, there must be a trusted multi-level network component that either makes the lower level keys available to the higher level members or acts as a gateway that decrypts an incoming message with a lower level key, reencrypts the message with a higher level key, before relaying the message. Alternatively, low level users may write up using a high level public key.

There can be issues of covert channel where lower level members can observe the existence of high level network activities. One solution is to use relays to confuse observers. For example, if messages coming out of SRI go through a relay that reencrypts them with a key it shares with the IBM relay, then all a third party can deduce is that someone at SRI sends a message to someone at IBM. Given a sufficiently random communication pattern between SRI and IBM, the bandwidth of this covert channel can be greatly reduced. In fact, the two relays can exchange useless messages at random intervals to further confuse the third party. It is unclear whether applications care about the threat of information leakage through covert channels, though similar techniques can be used to impede traffic analysis.

3.6 Two Trusted Multicast Facilities

Based on the principles and techniques discussed above, we have developed more specific architectures for two types of multicast: class-based multicast and mobile multicast.

In designing the class-based multicast, we take advantage of the following assumptions: (1) the network topology is relatively stable; (2) users belong to classes and this membership does not change often; and (3) a class is a broader concept than a group or session. One driving application environment is a corporation where an employee belongs to a department (which forms a class) and communication within the same department is largely open.

Our design of a secure mobile multicast is based the concept of a secure multicast agent and the (sealed) floating-letter-bottle analogy, which insulate security concerns from transport issues and mobility issues. The scenarios driving our design include a "flying doctor" initiating a multi-party consultation session, a business traveler taking part in a meeting being conducted at his company headquarters, and members of geologically separated departments of a corporation holding a joint meeting, all through multicast in (inter)networks such as today's Internet.

Due to page limit, we cannot provide details here. We plan to include them in a revised version of [9].

3.7 Related Work

One of the earliest work on process groups is [4]. In much of the later theoretical and practical work on broadcast protocols (e.g., [3]), the purpose is largely to achieve broadcast atomicity and/or to maintain causal or total ordering of messages, with no or little considerations for security.

When security is considered seriously, such as in secure reliable broadcast, secure atomic broadcast, or secure causal broadcast, the tendency is to introduce security into existing (and often complex) systems in a rather ad hoc manner. This results in security assumptions and architectures that are incompatible. A much better approach is to implement these complex broadcast protocols based on a trusted multicast facility. In this way, multiple applications share the same platform, have consistent security assumptions and protocols, and can thus coexist. This approach also greatly simplifies the (repetitive) task of getting the security mechanisms right in all applications.

Another related work is IP-Multicast [5, 6]. Our trusted multicast facility can be based on IP-Multicast and should not require modifications to IP-Multicast or lower layers of the network architecture (except for preventing misuse of network resources).

Prior efforts have addressed security issues in the Internet (e.g., [16, 7]). However, there were no architecture and protocols for secure multicasting. The Visa protocol controls the movement of datagrams (and thus the use of network resources) on a per user basis whereas in trusted multicast the granularity may need to be finer.

Heterogeneous Multicast (HMC) [14] explores the concept of multicast sessions in which users participate at different bandwidth levels. This concept can be extended into multicast at different security levels.

Due to the page limit, we are unable to give a more complete reference list.

4 Summary and Future Work

Advances in networking technologies have made multiparty communication an active area of research and development, where multicast is an important mode of communication as well as a good platform for building group-oriented services. In this paper, we have considered fundamental security issues in building a trusted multicast facility. We have discussed threats, new requirements for security, solutions, and some trade-offs between scalability and security. In particular, we have outlined protocols for secure session registration, for session membership change, and for secure and efficient broadcast, as well as architectures for class-based trusted multicast and for secure mobile multicast.

One area of future work is to prototype the technologies and protocols we have discussed in this paper. Such an experiment not only can bring valuable performance results, it can also establish a sound platform on which to build a wide range of more sophisticated protocols (e.g., for secure atomic broadcast and secure causal broadcast [11]) and trusted applications.

References

[1] M. Abadi and R.M. Needham. Prudent Engineering Practice for Cryptographic Protocols. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 122–136, Oakland, California, May 1994.

[2] T. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees – An Architecture for Scalable Inter-Domain Multicast Routing. In *Proceedings of ACM SIGCOMM*, pages 85–95, San Francisco, California, September 1993.

[3] K.P. Birman and T.A. Joseph. Reliable Communication in the Presence of Failures. *ACM Transactions on Computing Systems*, 5(1):47–76, February 1987.

[4] D.R. Cheriton and W. Zwaenepoel. Distributed Process Groups in the V Kernel. *ACM Transactions on Computer Systems*, 3(2):77–107, May 1985.

[5] S. Deering. Host Extensions for IP Multicasting. Request for Comments 1112, Internet Network Working Group, August 1989.

[6] S.E. Deering and D.R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[7] D. Estrin, J.C. Mogul, and G. Tsudik. Visa Protocols for Controlling Interorganizational Datagram Flow. *IEEE Journal on Selected Areas in Communications*, 7(4):486–498, May 1989.

[8] L. Gong. Authentication, Key Distribution, and Secure Broadcast in Computer Networks Using No Encryption or Decryption. Technical Report SRI-CSL-94-08, Computer Science Laboratory, SRI International, Menlo Park, California, May 1994.

[9] L. Gong and N. Shacham. Elements of Trusted Multicasting. Technical Report SRI-CSL-94-03, Computer Science Laboratory, SRI International, Menlo Park, California, March 1994.

[10] L. Gong and D.J. Wheeler. A Matrix Key Distribution Scheme. *Journal of Cryptology*, 2(2):51–59, 1990.

[11] M. Reiter and L. Gong. Preventing Denial and Forgery of Causal Relationships in Distributed Systems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 30–40, Oakland, California, May 1993.

[12] R.L. Rivest. The MD5 Message-Digest Algorithm. Request for Comments 1321, Internet Activities Board, April 1992.

[13] R.L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[14] N. Shacham. Multicast Routing of Hierarchical Data. In *Proceedings of the International Conference on Communications*, Chicago, Illinois, June 1992.

[15] C. Topolocic. Experimental Internet Stream Protocol, version 2 (ST-II). Request for Comments 1190, Internet Activities Board, October 1990.

[16] V.L. Voydock and S.T. Kent. Security Mechanisms in High-Level Network Protocols. *ACM Computing Surveys*, 15(2):135–171, June 1983.