A SURVEY OF MULTICAST SECURITY ISSUES AND ARCHITECTURES

Peter S. Kruus
Naval Research Labortory
Washington, DC 20375
kruus@itd.nrl.navy.mil

ABSTRACT

This paper addresses issues relevant to implementing security for IP multicast networks. These issues are of importance to application developers wishing to implement security services for their multicast applications. The paper investigates the steps required to create a secure multicast session including issues of group membership and key distribution. A common simple criteria is established that can be used to evaluate multicast keying architectures. The criteria focuses on the efficiency and scalability of the keying solution. Using this criteria, several keying architectures are evaluated and compared to determine their strengths and weaknesses.

INTRODUCTION

Multicast communications is an efficient means of distributing data to a group of participants. In contrast to unicast communications, multicast routing permits a single IP datagram to be routed to multiple hosts simultaneously. Membership in a multicast group is dynamic, allowing hosts to enter and leave the multicast session without the permission or knowledge of other hosts. The inherent benefits of multicast routing may also present some vulnerabilities making it susceptible to attack unless they are secured. The goal is to secure these vulnerabilities while maintaining the benefits of multicast service.

This paper presents issues relevant to securing IP multicast communications. An initial overview of multicast technology is presented followed by a general description of how security services can be applied within the scope of conventional multicast protocols. In many cases, cryptographic techniques such as encryption may be used to provide some of these security services. In this paper, issues related to the cryptographic keys supporting these techniques are shown to be crucial to the security of a multicast session. Many multicast security problems may be abstracted into a key distribution and management problem.

In order to secure a multicast session, a generic outline for multicast session registration and key distribution can be followed. Using this outline as a model, this paper establishes a set of criteria useful in evaluating several recently proposed multicast key distribution architectures. Each architecture focuses on methods designed to efficiently distribute keys to a multicast group. The techniques used to achieve this goal are different in each example.

OVERVIEW OF MULTICAST SERVICE

IP multicast is the transmission of IP datagrams to a host group [1]. A host group is a collection of multicast capable hosts that either transmit IP datagrams to or receive datagrams from a particular Class D IP address. Class D addresses are reserved specifically for multicast communications and can be dynamically assigned among multicast groups. Within the multicast group, membership is also dynamic. Hosts may enter and leave the group at will without permission from other group members. In a non-secure or public group, only knowledge of the multicast address is required for membership.

Several protocols are necessary to route IP datagrams to a multicast group. Hosts identify their desire to become part of a multicast group to their local router using the Internet Group Management Protocol (IGMP) messages defined in [1]. In order to deliver multicast IP datagrams to group members, routers may use one of several routing protocols that define the network routing topology [2, 3, 4, 24, 25]. The MBONE is an example of a multicast network overlaid on top of the traditionally unicast Internet by using the

Distance Vector Multicast Routing Protocol (DVMRP) [2, 5]. Some properties of these routing topologies may prove beneficial in multicast key distribution architectures. For example, in [7], Ballardie describes a key distribution architecture centered around a core router defined for the Core Based Tree (CBT) multicast routing protocol [4].

The scope of a multicast group can be limited by restricting the routing of its IP datagrams. By manipulating the *time-to-live* field in each IP datagram [6], hosts can limit the scope of their traffic by controlling the number of hops a datagram travels before routers discard it. By restricting the time-to-live field of a datagram, we create basic form of confidentiality for the group by limiting the potential audience of the data. This may be considered at best a very weak form of confidentiality that is difficult to enforce. Therefore, stronger mechanisms are required if we want greater assurance.

Multicast application layer data is typically encapsulated by the transport layer UDP protocol. The combination of UDP and IP protocols create an unreliable datagram service without error correction capabilities. Protocols such as the Real-Time Transport (RTP) protocol are designed to correct some of these deficiencies [8]. RTP runs on top of UDP and the underlying network protocol to provide end-to-end network transport functions for multicast audio and video conferences or sessions. A *session* is defined as the exchange of multimedia data (e.g., an audio conference) within a multicast group [9]. Several sessions may be active within a single multicast group (e.g., one for audio and another for video). The type of host participation in a multicast session can further define the nature of the session. In a *many-to-many* type application, multiple group members may receive and transmit data simultaneously during the session. In contrast, a *one-to-many* or *push* application typically has only one transmitter and many receivers for the session. The type of application may influence the design of a security architecture. For example, a one-to-many application is inherently centralized and may benefit from a security architecture centered around a single trusted host. Distributed many-to-many applications may benefit from distributed security architectures with multiple trusted hosts performing registration and key distribution functions.

Multicast sessions may also be described in terms of their membership. In general, a session is defined as either public or private. Both types are defined by the level of access control required to join the multicast group [10]. Public sessions are typically encountered on the MBONE and are supported by the dynamic nature of multicast communications (i.e., knowledge of the multicast address is the only requirement for membership). We can further restrict public sessions by requiring users to register and pass an authentication check in order to participate. In order to limit the scope of a private multicast session, both registration and authentication are required for participation [10]. To limit the visibility of the secure session, the session traffic is usually encrypted. In this paper, we define a *secure multicast session* as a private session with encryption.

Multicast applications can benefit from the addition of security services. Commercial applications that use public networks can limit user access to services and control user participation. Without these security features, user participation cannot be tightly controlled. Access control mechanisms applied during registration can limit participation to only paying customers. Military applications such as command and control have obvious benefits from the application of security. By tightly controlling access during registration, only users with the proper credentials can join the session. In both arenas, access control is the important initial step in defining the multicast group. Once the group is established, we can argue that most of our security concerns can be abstracted into a key management problem.

APPLYING SECURITY TO MULTICAST

Threats to IP multicast communications are similar to those for unicast IP transmissions. In general, threats include eavesdropping, the unauthorized creation of data, the unauthorized alteration of data, the unauthorized destruction of data, denial of service, and illegitimate use of data [11]. In the

case of multicast traffic, because of the inherent broad scope of a multicast session, the potential for attacks may be greater than for unicast traffic.

We can secure a system against these threats through the application of several fundamental security services (e.g., authentication, integrity, confidentiality). The security policy governing the system determines how these security services should be implemented in order to counter the threats. Implementation conflicts may arise when overlapping security policies cover a multicast group [12]. For example, conflicting policies may arise between entities separated by international boundaries. In this situation, the conflicting security policies might dictate using different encryption algorithms and key lengths. For this reason, security protocols used in multicast applications should be flexible to support a variety of security mechanisms. The IP security protocols defined for the Internet in [13], including the Encapsulating Security Payload (ESP) [14] and the Authentication Header (AH) [15], support this philosophy. These protocols are not restricted to a specific cryptographic algorithm or other security standard.

Fundamental Security Services

In order to counter the common threats to multicast communications, we can apply several of the fundamental security services, including authentication, integrity, and confidentiality as defined in [11].

Authentication services provide assurance of a host's identity. Authentication mechanisms can be applied to several aspects of multicast communications. Foremost, authentication is an essential part in providing access control to a secure multicast group. Applying authentication mechanisms to the registration process ensures that only authorized hosts are permitted to join the secure group. If the group employs cryptographic techniques such as encryption for security, then authentication measures may restrict access to the keys used to secure the group's communications. Group membership is essentially defined by access to these keys. Therefore, their availability must be restricted to only authorized group members.

In order to identify the source of multicast traffic, authentication mechanisms may also be applied directly to each IP datagram. This application serves to further define group membership by positively identifying each member of the group. Protocols such as AH provide authentication for IP datagrams and may be used for host authentication. Authentication is also an essential part of any key distribution protocol [16]. Because of the sensitive nature of keying material, authentication mechanisms can identify the source of the key material and provide a means to counter various masquerade and replay attacks that may be launched against a key distribution protocol.

Only strong authentication mechanisms are recommended for secure multicast applications. Digital signatures schemes, such as the Digital Signature Standard (DSS), are an examples of a strong authentication mechanisms based on public key technology [16]. In order to bind the identity of a host to their public key, certificates are formed that are digitally signed by a trusted certificate authority (CA). This process provides the necessary assurance to enable the proper identification of hosts during the registration process.

Integrity services provide assurance that multicast traffic is not altered during transmission. Integrity is not inherent to IP datagram traffic and is usually reserved for transport layer protocols (e.g., TCP). The lack of integrity services in IP can lead to spoofing attacks [17]. Integrity can be applied indirectly at the network layer with security protocols such as ESP and AH. In some applications where corrupted data can easily be detected (e.g., voice applications), this service is not vital. However, in other applications including key management protocols, integrity services are essential means of countering spoofing attacks.

Confidentiality services are essential in creating a private multicast session. Although encryption is typically used to provide this service, a weaker form of confidentiality may be achieved by limiting

the routing of session IP datagrams. Encryption can be applied at several layers of the protocol stack while maintaining the end-to-end service we desire. Transport protocols such as RTP support encryption mechanisms within their protocol definition. At the network layer, ESP provides confidentiality services for IP datagrams through encryption. Confidentiality services should also be applied to key management transactions during the exchange of key material. Key management protocols, such as the Internet Security Association and Key Management Protocol (ISAKMP) [16], support confidentiality services for key exchanges. Confidentiality may also be applied to session announcements allowing them to be advertised publicly while keeping the details of the session private.

Figure 1 presents an example implementation that summarizes some of these security concepts. In this example, each host creates a public key pair (i.e., k_x,k_x^{-1}). The private key of the pair (i.e., k_x^{-1}) is kept secret by the host and is used to sign messages and key material (i.e., if the host performs key distribution functions). This signature uniquely identifies the host to other group members. The public key of the pair (e.g., k_x) is signed by the CA and distributed to other group members in the form of a certificate (e.g., CA<X>). Hosts can verify the digital signatures of other group members using the public key found in their certificate. In this example, host A has generated and distributed a group key, K_S, to the multicast group. After hosts are authenticated by host A, the signed group key is securely distributed to valid group members (i.e., host B and C). K_S defines the secure group. Host D is not a valid member and therefore is not issued a copy of K_S. To further define group membership, multicast messages encrypted in K_S are signed by group members using their private key. Figure 1 shows the message *m* encrypted by the group key K_S and signed by the transmitting host B (i.e., with k_b^{-1}). Using K_S and CA<B>, group members A and C can decrypt the message and verify its origin.

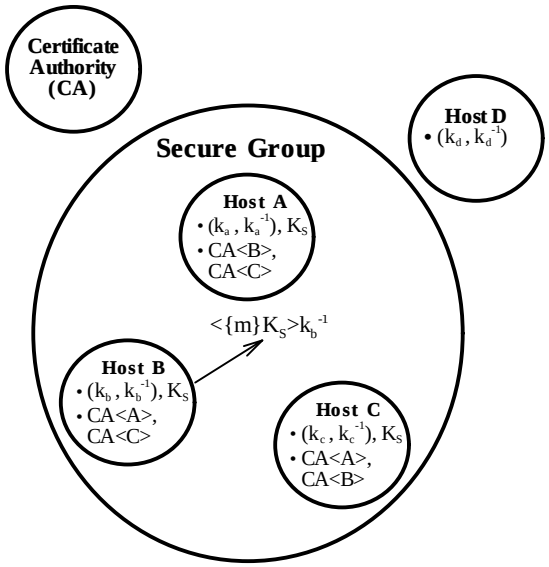

Figure 1. Hosts within a Secure Group

Communicating Security Requirements

After the security services required for a multicast session have been identified, it is important to communicate the details of their implementation to current and potential group members. This may include information about the type of cryptographic algorithm, the length of a crypto-period, key length, type of authentication mechanisms used, and other security related information describing the implementation details of a particular secure session. In some unicast environments, these parameters may be negotiated between hosts. ISAKMP supports the negotiation of security parameters between two hosts. This feature may be beneficial in situations of conflicting security policies. However, because of the potentially large number of participants in a multicast group, it is generally not efficient to allow the negotiation of security parameters. Therefore, session requirements are typically defined by the *initiator* of the session and announced to potential participants. The initiator may record the security parameters for a secure session in the form of a *security association* (SA) [13]. It is possible to have multiple SAs within a secure group. For example, audio traffic may be encrypted under one key and video under another key with each session described by a separate SA. When using ESP, a *security parameter index (SPI)* within each IP

datagram identifies the SA required to decrypt the traffic. In all cases, the details of a SA must be known by group members prior to the start of a secure session.

The initiator of a session may distribute the SA for a secure session prior to the session start time using techniques similar to those used to announce non-security parameters (e.g., session start time, type of session protocol used). Two methods are typically used to announce a session: advertisement and invitation.

The initiator may advertise their desire to start a session by using the Session Announcement Protocol (SAP) [9]. The announcement may be advertised to potential members by broadcasting the announcement to a particular multicast address reserved for receiving session announcements. Alternately, announcements can be extended to a more selective group through invitation protocols such as the Session Initiation Protocol (SIP) [18] which directly contacts potential participants. Both SAP and SIP use the Session Description Protocol (SDP) [19] to describe the requirements for the multicast session. The defined requirements may include both security and non-security parameters. These announcement techniques offer an efficient means to distribute the SA and other non-security conference information to potential participants prior to the start of the secure session.

Key Management Issues for Multicast

Through the use of encryption and digital signatures, we can achieve desired levels of confidentiality, integrity, and authentication. Assuming that we use strong security mechanisms that when implemented properly cannot be broken by frivolous cryptanalytic attacks, we can focus our security concerns on protecting the key material. We may generally assume that our cryptographic algorithms cannot be broken and thus, all security resides in the key material. Therefore, we focus our security concerns around key management, key distribution, and access control for key material. With this in mind, a secure multicast session can be defined by its Class D IP address and the required keying material for the session.

The size, type (e.g., asymmetric vs. symmetric), and number of keys required to secure a multicast session is determined by the encryption mechanism employed and the keying architecture. In general, session participants may use a common *group traffic encryption key (GTEK)* to encrypt session data. The initiator of the session may use *group key-encryption-keys (GKEKs)* to encrypt future session keys (i.e., GTEKs) for distribution to group members. For a private multicast sessions, access to these keys must be restricted to maintain the security of the overall session. Therefore, during the registration process it is necessary to require strong authentication mechanisms to establish the identity of potential participants prior to distributing key material. The specific access control mechanism may be unique to the application. For example, identity based access control mechanisms may be appropriate for some commercial applications while military models may use permission based schemes that identify a participant's clearance level. When these personal attributes are bound to a signed certificate, the identify of a participant and their assigned permissions may be verified by the certificate's digital signature and its relationship in a certificate hierarchy [20].

Depending on a system's security policy and the amount of traffic encrypted under a particular key, it may be necessary to periodically issue a new key or "rekey" a multicast session. A rekey may also be required in the event of a key compromise. In this case, it is necessary to exclude the compromised site from future communications. Therefore, a rekey must be targeted to specifically exclude the compromised site while retaining the rest of the group membership. Depending on the security policy in place, the definition of a compromise might include the voluntary exit of a participant from a secure session. If this occurs, the entire group may require a rekey in order to prevent the excluded participant from rejoining the group at a later time without re-registering for the session. In addition, the keying architecture should

prevent collusion by a group of disbanded members from generating or recreating the new group key.

THE SECURE MULTICAST PROCESS

As described in [12], a common process may be applied to the establishment and maintenance of a secure multicast session regardless of the implementation details of the supporting keying scheme. Common functions may be derived from the registration and authentication processes required for private multicast sessions. In the following example, we assume that authentication services are based on a public key certificate hierarchy. A certificate authority (CA) serves as a trusted and centralized authority for participant identification. All participants have access to the CA for verification of digital signatures. The following steps provide an overview of the initialization, registration, and maintenance processes required for secure multicast sessions:

1. Someone identifies the need for a secure session (i.e., the *Initiator*).

2. After the need for the secure session has been established, the Initiator defines the parameters required to participate in the session using a description protocol such as SDP. Parameters include security related details that may be defined in a security association (SA) as well as other non-security related parameters (e.g., session start time, session address).

3. Prior to the start of the secure session, the Initiator determines whether assistance is required to perform the participant registration or key distribution functions. The identity and location of other trusted entities assisting in these functions can be recorded in the session description. This allows participants to directly contact the appropriate trusted entity at the start of the session. In some keying architectures, such as the one described in [7] for CBT multicast routing networks, keying and registration functions are immediately passed off to a trusted network "core". Other architectures delegate responsibilities to outside trusted entities only as needed [21]. In a truly distributed keying architecture such as that described in [10], delegation of registration and key distribution functions to all active participants is assumed.

4. In order to enlist membership for the secure session, the session description is announced to potential participants. The announcement may come in the form of a posted advertisement or a personal invitation to each group member. The Initiator may list invited participants in a session access control list (ACL). The ACL should be held by all authorized keying entities. Updates to the ACL must be distributed among these trusted entities. Confidentiality may be applied to session announcements in order to conceal the existence of a secure session. In most cases, the Initiator may use multicast techniques to efficiently announce the session to all potential participants prior to its start.

5. After receiving the session announcement, potential participants may register for the secure session if they can meet its requirements. Registration is performed by the Initiator or other trusted entity. As part of the registration process, participant identities and their permissions are authenticated. Only valid participants are extended the invitation to become members of the secure group and are issued the required keying material for the session. Keys exchanged during the registration process must be provided the basic security services. Depending on the security policy governing the session, it may be required to provide a membership list of all registered participants to the group. After reviewing the membership list, some participants may decide to not participate further in the session. In this case, the security policy for the session may dictate that participants delete the session key when exiting the session.

6. During the course of a secure session, it may be necessary to perform several maintenance operations including keying and registration functions. In order to support the dynamic nature of multicast communications, it may be required to add or delete participants after the start of the session. In some applications, the percent of membership roll-over may be limited throughout the lifetime of the session [12]. Participants added during an ongoing session must complete the

authenticated registration process prior to receiving group key material. Individual or groups of participants may be dropped from the secure session either cooperatively or non-cooperatively [21]. With a cooperative exit from the session, the participant voluntarily leaves the group and is asked to erase all session key material. Depending on the security policy governing the session, the keys held by the participant may be treated as compromised. In the case of a key compromise, a non-cooperative drop is required to forcefully remove the compromised participant from the secure session. To remove the participant, all potentially compromised keys must be replaced by performing a rekey. The rekey involves rekeying all affected participants with a new session key (e.g., K_S).

CRITERIA FOR SECURE MULTICAST

There are numerous issues to consider when developing a secure multicast application. [22] provides a general overview of non-security issues that developers should consider. This paper has focused on identifying security related issues developers should consider when implementing private multicast sessions. Several of these issues relate to the key material used to secure the multicast traffic. Foremost, the registration process provides a level of access control for key material. For a private session, all participants should be authenticated during the registration process. Upon successful registration, key material and group membership may be extended. For a secure session, group membership is defined by the session's IP multicast address and the key material required to communicate securely with other group members.

A complete solution to the multicast security problem addresses all relevant security issues including session announcement, registration, etc. As mentioned previously, we may reduce many of our security concerns to a key management and distribution problem. This enables the efficient comparison and evaluation of various solutions to the secure multicast problem. In order to make a fair comparison, we define a limited set of criteria common to all viable solutions. The criteria helps to determine the advantages and disadvantages of each keying architecture by focusing on key distribution efficiency and the overall scalability of the architecture. Other issues including participant registration and authentication are not considered but are equally important.

Criteria	
Efficiency: Initial Keying	What is the efficiency of the initial key distribution exchange at the start of the session ?
Efficiency: Rekeying	What is the efficiency of rekey operations (i.e., for reasons of compromise or crypto-period roll-over)?
Computation Requirements	What level of computational resources is required by the key distributor and members to process keying messages?
Storage Requirements	What amount of storage is required by participants for key storage?
Scalability	Is the solution scalable for large and small groups (i.e., large > 100 members)?

Table 1. Criteria for Secure Multicast

Several solutions to the multicast key distribution problem have been presented in papers such as [7, 10, 12, 21, 23]. The core criteria used to evaluate several of these proposed multicast keying architectures is shown in table 1. This criteria focuses on key distribution efficiency, the ability to support dynamic user entry and exit in an already established secure session, computation requirements placed on the system to perform the keying operations, key storage requirements, and scalability. Efficiency is recorded in *big-O* notation as a measure of the number of key related messages transmitted per operation (e.g., rekey, initial keying). The size of each message is also considered. However, it is assumed that for some

applications network throughput will increase over time making the issue of size less important. Although efficiency is of primary concern to networks with limited bandwidth, it also provides a convenient measure of system performance.

KEY DISTRIBUTION ARCHITECTURES

In order to improve the efficiency of a keying solution for secure multicast applications, it is often beneficial to use features of multicast communications that makes it an efficient form of group communications. The ideal key distribution efficiency in a multicast environment is O(1). In such a scenario, a centralized server may transmit only a single keying message to the entire group to perform a group rekey. Every group member can extract the required key material from this one message. In contrast, the efficiency of using unicast techniques to distribute a group key separately to each group member is O(n). Note, in most cases it is more efficient to perform the initial keying of participants in a unicast fashion during the registration process. The registration function is inherently a one-to-one between a single participant and the Initiator or other trusted registration authority. By coupling registration with key distribution, the overall number of transmissions required to perform both functions can be reduced.

Keying functions may be either centralized or distributed throughout the architecture. In a centralized architecture, keying functions are restricted to a single trusted authority. In some cases, this may be the Initiator of a session or another entity assigned by the Initiator to handle these vital functions. For scalability purposes, keying and registration functions may be distributed to other trusted entities. Applications that are of the type “one-to-many” may benefit from a strictly centralized architecture. Alternatively, distributed architectures may prove more scalable since processing and storage requirements are distributed across the network.

The following sections provide a brief analysis of several key distribution architectures [7, 10, 12, 21, 23]. Each section presents a brief description of the architecture followed by an analysis of its performance. Performance is measured against the criteria established in the previous section. We ignore issues of session definition, announcement, and registration, focusing instead on the key related criteria.

Manual Key Distribution

As noted in [12], manual key distribution architectures are easily understood by users and in many cases already in place (e.g., the military). In this type of architecture, all key generation and distribution functions reside at a central key distribution center (KDC). Key material requirements for a secure multicast session must be determined by the Initiator well enough in advance for the KDC to generate and manually distribute the keys to all participants. There is no computational load on individual participants and storage requirements may be limited to GTEK and GKEK storage.

Because significant off-line coordination is required with the KDC, this solution is not scalable. Also, it is generally slow to respond to dynamic user entries and exits from the secure multicast group. Manual keying techniques are also slow to respond to compromise. In the event of a group key compromise, new key material must be distributed manually to valid participants.

Pairwise Keying

Several keying architectures have been designed around the concept of pairwise keys [7, 12, 21]. With this type of architecture, the session Initiator distributes the required key material for the secure session. The Initiator may also perform registration and authentication functions or pass them to another trusted entity. Keying function may also be distributed among trusted entities. During the registration process, the Initiator establishes and caches a unique session key with each participant creating a “pairwise” keying relationship between the Initiator and participant. These unique participant session keys are used to encrypt group keys for each authenticated participant. The encrypted keys can be distributed

to each participant individually or multicast in a single complete message containing all of the individually encrypted keys.

In [7], Ballardie proposed a pairwise multicast key distribution solution based on the Core Based Tree (CBT) multicast routing protocol. In a CBT architecture, a routing tree is centralized around a *core* router. The core router is centralized for the multicast group making it a natural authorization and key distribution point. As other routers join the tree, they may undergo an authentication process with the core. Through the addition of other trusted routers to the tree, keying functions may be distributed outside the core router allowing the solution to scale to large groups.

In the CBT architecture, the Initiator of the secure session creates an access control list (ACL) and security association (SA) for the session. The ACL and SA are passed to the core who then creates a GTEK and GKEK for the session. The core distributes the ACL, GTEK, and GKEK to secondary routers as they are authenticated and added to the tree. Ballardie recommends using a keying protocol such as ISAKMP to distribute keys between group members and the trusted routers. In a pairwise fashion, ISAKMP enables the creation of a unique session key between the two entities that can be used to securely exchange the group key material. The computational requirements at each end of this exchange are limited to the creation of the unique session key.

Because each participant must be keyed individually, the efficiency of the initial keying transmissions for the pairwise approach is O(n). Each participant must create a pairwise key between with the Initiator prior to receiving the group key material. Several pairwise architectures including Ballardie's scheme and the Group Key Management Protocol (GKMP) described in [21] attempt to distribute keying functions to other trusted entities. Caching of the unique participant session keys is not required in these architectures; however, their presence makes the rekey operation much more efficient. Otherwise, in order to rekey the group with a new group key requires essentially the creation of another secure group that excludes untrusted participants. In [21], the use of a GKEK can greatly improve the efficiency of a rekey. However, if the GKEK is compromised, the efficiency reverts back to the efficiency of creating a new group. Therefore, like the initial keying functions, the rekey function has an efficiency of O(n).

Hierarchical Trees

In [12], Wallner, et al propose a hierarchical keying scheme that attempts to satisfy the problem of rekeying to disenroll participants from a secure group. A hierarchical tree of *key-encryption-keys (KEKs)* is created with the GTEK used for the encryption of multicast traffic residing at the root of the tree. Participants become leaves of the tree with each having their own unique KEK. Each tier above a participant corresponds to a different KEK. Participants store all of the keys within the tree in a path between the themselves and the root. In the event of a compromise, the Initiator may use the tiered KEKs to exclude a single participant or groups of participants during a rekey. All compromised KEKs and the GTEK are replaced during the rekey process.

In the event of a compromise, the number of transmission required to rekey the affected participants in the tree is on the order of O(logn). The number of messages required to rekey the tree is $(k-1)d$ for a k-ary tree of depth[1] d. Key storage requirements for each participant is on the order of $d+1$ keys. There Initiator must store all keys in the tree.

Efficiency in this scheme is achieved by using a *divide-and-conquer* algorithm on the k-ary tree. While multiple messages may be required to rekey the compromised section of the tree, other less affect sections of the tree can be rekeyed more efficiently with fewer and smaller messages. This feature makes the scheme scalable towards large groups.

[1] Rekey messages transmitted to each participant may contain multiple keys.

In the hierarchical tree scheme, the keying computational requirements are greatest at the Initiator site. During the initial keying of the tree, the Initiator must generate a unique KEK with each participant in the same fashion as the pairwise approach (i.e., O(n)). In addition, the Initiator must also generate keys for all other tiers of the tree. During a rekey, the Initiator must encrypt the new group key in the appropriate sequence of KEKs corresponding to the tree hierarchy.

Secure Lock

The secure lock described by Chiou and Chen in [23] utilizes the efficiency of multicast communications to key and rekey session participants. Using the Chinese Remainder Theorem (CRT), a secure lock is constructed that is used to "lock" the deciphering group session key. The single lock is transmitted with each encrypted message. Only users in the secure group can "unlock" the session key.

The principle behind the secure lock lies in the mathematics of the CRT. The CRT states that for N_1, N_2, ..., N_n positive, relatively prime integers and R_1, R_2, ...,R_n positive integers, a set of congruous equations

$X= R_1 \text{mod} N_1$ $X= R_2 \text{mod} N_2$ $X= R_N \text{mod} N_N$

have a common solution X in the range of [1, L-1] where $L=N_1*N_2*N_3*...N_n$ and *n* is the number of participants in the group [23]. Chiou and Chen use the CRT properties to generate X where $R_i = E_{eki}(d)$ is the session key *d* encrypted by the function *E* using participant u_i*'s* public enciphering key *eki* (part of a public key pair). The common lock X is generated by the Initiator using each participant's public enciphering key. Each participant can recover the locked session key *d* by applying the CRT. The participant computes *d* using their secret deciphering key *dki*. Only participants whose enciphering keys were included in the calculation of X can unlock *d.*

The secure lock method is flexible towards the dynamic addition and deletion of group participants. Using the CRT, the Initiator can generate the common X and rekey the group to include or exclude certain participants from future communications. Only those participants whose *eki* was used in the computation of X can recover the session key. Because X is common among all valid participants, the efficiency of the transmission of the lock is O(1). Storage requirements at each participant site is limited to their public key pair. In order for the Initiator to create the lock, it must store the public keys for each participant.

As noted in [23], computation time of the common X using CRT is restrictive and may only be efficient for small groups. However, the decipherment of *d* by each participant is fairly efficient. The overall efficiency of the distribution of the secure lock (i.e., O(1)) may be eclipsed by its computation time for large groups. Additionally, since the computation of a common X is not distributable, the scheme is inherently centralized. Therefore, the secure lock scheme does not scale well to large groups unless a method for distributing the generation of X can be defined.

Distributed Registration and Key Distribution (DiRK)

Distributed Registration and Key Distribution (DiRK) [10] by Oppliger and Albanese is a key distribution protocol designed for application over the MBONE. It distinguishes between active and passive participants in a multicast session. In a truly distributed fashion, any active participant may assist the Initiator with registration and key distribution duties.

During the initialization phase, the Initiator generates a session public key pair (k_a, k_a^{-1}). The Initiator announces the session with a signed copy of the public key k_a. The private key k_a^{-1} is used by the Initiator to sign subsequent messages. The Initiator also generates a group key for the session, K_S.

After receiving the session announcement, hosts may request to join the group by sending a *registration request* message to the multicast group (i.e., transmitted to the group's Class D IP address). The request message contains the public

part of a public key pair created by the participant for this session (i.e., k_x). Any active participant already keyed with K_S may respond to valid users with the group key K_S encrypted in k_x. Timers and the use of the IP time-to-live field can be used to restrict responses to *registration requests* to only locally active participants and thus prevent a flood of messages from the group. Only user X can decrypt the group key encrypted in k_x. The response also contains a certificate signed by the responder that validates the returned key. The returned certificates form a hierarchical participant registration tree (PRT) that can be used to trace a certificate back to the Initiator.

DiRK includes a *registration validation* message to keep participants up-to-date on the current group membership. Participants periodically send validation messages to the group that include a copy of the participants public session key (i.e., k_x) and the certificate they received when they were registered for the session.

Several rekey options are supported by DiRK. A *full session* rekey can be performed by the Initiator to rekey all participants with a new key. This rekey protocol simply encrypts the new key in the previous session key. In the event of a compromise, a selective rekey can be performed by the Initiator to rekey only selective participants. The Initiator uses the public session keys received in the registration validation messages to individually encrypt the new session key for each valid participant. A single message is then sent to the group with the individually encrypted keys. Unique to DiRK is a *distributed session rekey* message that permits active participants to rekey the group. The initial distributed session rekey message contains a list of participants banned from participation. This list is examined by active participants before issuing the new group key to other potential participants.

The overall architecture of DiRK is very efficient because of its distributed nature. The processing required to key and rekey *n* participants (i.e., O(n)) is distributed across the network to all active participants. Although the selective keying function is O(1), this size of the rekey message is the size of n.

DiRK is highly scalable because of its distributed properties. However, these same properties also distribute the trusted registration and authentication processes across the network. In order to validate messages, users must retain the certificates for all local active participants.

SUMMARY OF KEYING ARCHITECTURES

Each of the solutions presented in [7, 10, 12, 21, 23] attempts to efficiently solve the multicast key distribution problem in a slightly different fashion. It is important to note that the best solution for one particular application may not be well suited for another application. For example, centralized applications such as multicast video servers may benefit from a centralized keying architecture while distributed command and control applications may benefit more from an equally distributed key distribution scheme. The following paragraphs summarize the evaluation results from the previous section.

Manual keying methods were determined to be too slow for dynamic multicast sessions in which membership is not defined prior to the start of the session. However, in some military environments with a well structured manual key distribution architecture already in place, this solution may be the easiest to implement.

Pairwise keying techniques typically provide linear efficiency for initial keying and rekey operations. By consolidating all rekey messages into a single multicast message, the efficiency of the rekey can be dramatically improved. However, this technique increases the overall size of the rekey message to *n*. Key storage requirements for pairwise techniques are minimal at participant sites but requires *n* keys to be stored with the key distributor. This method is scalable if keying and registration functions are distributed to other trusted entities.

The *hierarchical trees* method provides linear initial keying performance and improved logarithmic rekey performance. The size[2] of any rekey message is no greater than *(k-1)d*. Key storage requirements at each participant site is *d+1* keys while the Initiator must store all KEKs and the GTEKs. The solution is scalable because of the logarithmic rekey performance.

The *secure lock* method has linear initial keying performance and an impressive constant rekey performance. The size of the rekey message is also constant providing the best rekey performance of all methods reviewed. The drawbacks of this method include the computation time for the lock and the fact that the technique is inherently centralized and may not scale to large groups.

In order to improve overall system efficiency, *DiRK* distributes linear initial keying and rekey functions among active group members. However, a question of trust may arise because the registration and key distribution functions are distributed in such a broad fashion. Otherwise, the solution is scalable to large networks.

CONCLUSION

Multicast sessions can be secured through the application of several fundamental security services. A complete solution to the multicast security problem addresses all aspects of the application of these security services. This paper has shown that many of these security concerns can be abstracted into a key management problem. The key material required to communicate successfully and securely within a session defines the secure group. Therefore, access to group key material must be restricted and tightly controlled.

By using a set of criteria focusing on key related issues, this paper analyzed several different keying solutions. Because the best solution for one particular application might not be well suited for another application, it is important to fully understand the requirements and security policy of the application prior to applying a security solution. In general, it has been observed that most keying solutions follow a similar process for securing a multicast session. In some cases, by combining the inherently linear registration and initial keying functions together into a single step, the overall efficiency of the keying scheme can be improved.

A security solution should compliment rather than drive the implementation of a multicast application. The application of security should be transparent to the user and work efficiently with other required protocols. Future work should focus on achieving a truly integrated security solution that functions together with other non-security functions and existing multicast protocols.

[2] For a k-ary tree of depth d.

REFERENCES

[1] *Host Extensions for IP Multicast*, S. Deering, RFC 1112, 1989.

[2] *Distance Vector Multicast Routing Protocol*, S. Deering, C. Partridge, and D. Waitzman, RFC- 1075, 1 November 1988.

[3] *Multicast Extensions to OSPF*, J. Moy, RFC 1584, Proteon, Inc., March 1994.

[4] *Core Based Trees (CBT) Multicast Routing Architecture*, A. Ballardie, Internet-Draft, May 1997.

[5] *Routing in the Internet*, C. Huitema, Prentice Hall, 1995.

[6] *Internet Protocol*, J. Postel, , RFC-791, USC/Information Sciences Institute, September 1981.

[7] *Scalable Multicast Key Distribution*, A. Ballardie, RFC-1949, May 1996.

[8] *RTP: A Transport Protocol for Real-Time Applications*, H. Schulzrinne et al, RFC-1889, January 1996.

[9] *SAP: Session Announcement Protocol*, M. Handley, Internet-Draft, 19 November 1996.

[10] *Distributed Registration and Key Distribution (DiRK)*, R. Oppliger and A. Albanese, Proceedings of the 12th International Conference on Information Security (IFIP SEC '96), Island of Samos (Greece), May 21-24, 1996, Chapman & Hall, London, pp. 199-208.

[11] *Computer Communications Security: Principles, Standard Protocols and Techniques*, W. Ford, Prentice Hall, 1994.

[12] *Key Management for Multicast: Issues and Architecture*, D. Wallner, E. Harder, and R. Agee, Internet-Draft, draft-wallner-key-arch-00.txt, 1 July 1997.

[13] *Security Architecture for the Internet Protocol*, R. Atkinson, RFC-1825, Naval Research Laboratory, August 1995.

[14] *IP Encapsulating Security Payload (ESP)*, R. Atkinson, RFC-1827, Naval Research Laboratory, August 1995.

[15] *IP Authentication Header*, R. Atkinson, RFC- 1826, Naval Research Laboratory, August 1995.

[16] *Internet Security Association and Key Management Protocol (ISAKMP)*, D. Maughan, M. Schertler, M. Schneider, J. Turner, Internet-Draft, 21 February 1997.

[17] *Security Problems in the TCP/IP Protocol Suite*, S. Bellovin, ACM Computer Communications Review, Vol. 19, No. 2, March 1989.

[18] *SIP: Session Initiation Protocol*, Handley, Schulzrinne, Schooler, Internet-Draft, 31 July 1997.

[19] *SDP: Session Description Protocol*, M. Handley and V. Jacobson, Internet-Draft, 2 September 1997.

[20] *Applied Cryptography, Second Edition: Protocols, Algorithms and Source Code in C*, B. Schneier, John Wiley & Sons, Inc., 1996.

[21] *Group Key Management Protocol (GKMP) Architecture*, H. Harney and C. Muckenhirn, RFC-2094, July 1997.

[22] *Taxonomy of Communication Requirements for Large-scale Multicast Applications*, R. Briscoe and P. Bagnall, Internet-Draft, 29 July 1997.

[23] *Secure Broadcasting Using the Secure Lock*, G.H. Chiou and W.T. Chen, IEEE Transactions on Software Engineering, v. SE-15, n. 8, August 1989, pp. 929-934.

[24] *Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification*, D. Estrin, et al, Internet-Draft, 15 March 1997.

[25] *Protocol Independent Multicast Version 2: Dense Mode Specification*, D. Estrin, et al, Internet Draft, 2 April 1997.