Network Working Group
Request for Comments: 2627
Category: Informational

D. Wallner
E. Harder
R. Agee
National Security Agency

June 1999

Key Management for Multicast: Issues and Architectures

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This report contains a discussion of the difficult problem of key management for multicast communication sessions. It focuses on two main areas of concern with respect to key management, which are, initializing the multicast group with a common net key and rekeying the multicast group. A rekey may be necessary upon the compromise of a user or for other reasons (e.g., periodic rekey). In particular, this report identifies a technique which allows for secure compromise recovery, while also being robust against collusion of excluded users. This is one important feature of multicast key management which has not been addressed in detail by most other multicast key management proposals [1,2,4]. The benefits of this proposed technique are that it minimizes the number of transmissions required to rekey the multicast group and it imposes minimal storage requirements on the multicast group.

1.0 MOTIVATION

It is recognized that future networks will have requirements that will strain the capabilities of current key management architectures. One of these requirements will be the secure multicast requirement. The need for high bandwidth, very dynamic secure multicast communications is increasingly evident in a wide variety of commercial, government, and Internet communities. Specifically, the secure multicast requirement is the necessity for multiple users who share the same security attributes and communication requirements to securely communicate with every other member of the multicast group using a common multicast group net key. The largest benefit of the multicast communication being that multiple receivers simultaneously get the same transmission. Thus the problem is enabling each user to determine/obtain the same net key without permitting unauthorized parties to do likewise (initializing the multicast group) and securely rekeying the users of the multicast group when necessary. At first glance, this may not appear to be any different than current key management scenarios. This paper will show, however, that future multicast scenarios will have very divergent and dynamically changing requirements which will make it very challenging from a key management perspective to address.

2.0 INTRODUCTION

The networks of the future will be able to support gigabit bandwidths for individual users, to large groups of users. These users will possess various quality of service options and multimedia applications that include video, voice, and data, all on the same network backbone. The desire to create small groups of users all interconnected and capable of communicating with each other, but who are securely isolated from all other users on the network is being expressed strongly by users in a variety of communities.

The key management infrastructure must support bandwidths ranging from kilobits/second to gigabits/second, handle a range of multicast group sizes, and be flexible enough for example to handle such communications environments as wireless and mobile technologies. In addition to these performance and communications requirements, the security requirements of different scenarios are also wide ranging. It is required that users can be added and removed securely and efficiently, both individually and in bulk. The system must be resistant to compromise, insofar as users who have been dropped should not be able to read any subsequent traffic, even if they share their secret information. The costs we seek to minimize are time required for setup, storage space for each end user, and total number of transmissions required for setup, rekey and maintenance. It is also envisioned that any proposed multicast security mechanisms will be implemented no lower than any layer with the characteristics of the network layer of the protocol stack. Bandwidth efficiency for any key management system must also be considered. The trade-off between security and performance of the entire multicast session establishment will be discussed in further detail later in this document.

The following section will explain several potential scenarios where multicast capabilities may be needed, and quantify their requirements from both a performance and security perspective. It will be followed in Section 4.0 by a list of factors one must consider when designing a potential solution. While there are several security services that will be covered at some point in this document, much of the focus of this document has been on the generation and distribution of multicast group net keys. It is assumed that all potential multicast participants either through some manual or automated, centralized or decentralized mechanism have received initialization keying material (e.g. certificates). This document does not address the initialization key distribution issue. Section 5 will then detail several potential multicast key management architectures, manual (symmetric) and public key based (asymmetric), and highlight their relative advantages and disadvantages (Note:The list of advantages and disadvantages is by no means all inclusive.). In particular, this section emphasizes our technique which allows for secure compromise recovery.

3.0 MULTICAST SCENARIOS

There are a variety of potential scenarios that may stress the key management infrastructure. These scenarios include, but are not limited to, wargaming, law enforcement, teleconferencing, command and control conferencing, disaster relief, and distributed computing. Potential performance and security requirements, particularly in terms of multicast groups that may be formed by these users for each scenario, consists of the potential multicast group sizes, initialization requirements (how fast do users need to be brought on-line), add/drop requirements (how fast a user needs to be added or deleted from the multicast group subsequent to initialization), size dynamics (the relative number of people joining/leaving these groups per given unit of time), top level security requirements, and miscellaneous special issues for each scenario. While some scenarios describe future secure multicast requirements, others have immediate security needs.

As examples, let us consider two scenarios, distributed gaming and teleconferencing.

Distributed gaming deals with the government's need to simulate a conflict scenario for the purposes of training and evaluation. In addition to actual communications equipment being used, this concept would include a massive interconnection of computer simulations containing, for example, video conferencing and image processing. Distributed gaming could be more demanding from a key management perspective than an actual scenario for several reasons. First, the nodes of the simulation net may be dispersed throughout the country.

Second, very large bandwidth communications, which enable the possibility for real time simulation capabilities, will drive the need to drop users in and out of the simulation quickly. This is potentially the most demanding scenario of any considered.

This scenario may involve group sizes of potentially 1000 or more participants, some of which may be collected in smaller subgroups. These groups must be initialized very rapidly, for example, in a ten second total initialization time. This scenario is also very demanding in that users may be required to be added or dropped from the group within one second. From a size dynamics perspective, we estimate that approximately ten percent of the group members may change over a one minute time period. Data rate requirements are broad, ranging from kilobits per second (simulating tactical users) to gigabits per second (multicast video). The distributed gaming scenario has a fairly thorough set of security requirements covering access control, user to user authentication, data confidentiality, and data integrity. It also must be "robust" which implies the need to handle noisy operating environments that are typical for some tactical devices. Finally, the notion of availability is applied to this scenario which implies that the communications network supplying the multicast capability must be up and functioning a specified percentage of the time.

The teleconference scenario may involve group sizes of potentially 1000 or more participants. These groups may take up to minutes to be initialized. This scenario is less demanding in that users may be required to be added or dropped from the group within seconds. From a size dynamics perspective, we estimate that approximately ten percent of the group members may change over a period of minutes. Data rate requirements are broad, ranging from kilobits per second to 100's of Mb per second. The teleconference scenario also has a fairly thorough set of security requirements covering access control, user to user authentication, data confidentiality, data integrity, and non-repudiation. The notion of availability is also applicable to this scenario. The time frame for when this scenario must be provided is now.

4.0 ARCHITECTURAL ISSUES

There are many factors that must be taken into account when developing the desired key management architecture. Important issues for key management architectures include level (strength) of security, cost, initializing the system, policy concerns, access control procedures, performance requirements and support mechanisms. In addition, issues particular to multicast groups include:

1. What are the security requirements of the group members? Most likely there will be some group controller, or controllers. Do the other members possess the same security requirements as the controller(s)?

2. Interdomain issues - When crossing from one "group domain" to another domain with a potentially different security policy, which policy is enforced? An example would be two users wishing to communicate, but having different cryptoperiods and/or key length policies.

3. How does the formation of the multicast group occur? Will the group controller initiate the user joining process, or will the users initiate when they join the formation of the multicast group?

4. How does one handle the case where certain group members have inferior processing capabilities which could delay the formation of the net key? Do these users delay the formation of the whole multicast group, or do they come on-line later enabling the remaining participants to be brought up more quickly?

5. One must minimize the number of bits required for multicast group net key distribution. This greatly impacts bandwidth limited equipments.

All of these and other issues need to be taken into account, along with the communication protocols that will be used which support the desired multicast capability. The next section addresses some of these issues and presents some candidate architectures that could be used to tackle the key management problem for multicasting.

5.0 CANDIDATE ARCHITECTURES

There are several basic functions that must be performed in order for a secure multicast session to occur. The order in which these functions will be performed, and the efficiency of the overall solution results from making trade-offs of the various factors listed above. Before looking at specific architectures, these basic functions will be outlined, along with some definition of terms that will be used in the representative architectures. These definitions and functions are as follows:

1. Someone determines the need for a multicast session, sets the security attributes for that particular session (e.g., classification levels of traffic, algorithms to be used, key variable bit lengths, etc.), and creates the group access control list which we will call the initial multicast group participant list. The entity which performs these functions will be called the INITIATOR. At this point, the multicast group participant list is strictly a list of users who the initiator wants to be in the multicast group.

2. The initiator determines who will control the multicast group. This controller will be called the ROOT (or equivalently the SERVER). Often, the initiator will become the root, but the possibility exists where this control may be passed off to someone other than the initiator. (Some key management architectures employ multiple roots, see [4].) The root's job is to perform the addition and deletion of group participants, perform user access control against the security attributes of that session, and distribute the traffic encryption key for the session which we will call the multicast group NET KEY. After initialization, the entity with the authority to accept or reject the addition of future group participants, or delete current group participants is called the LIST CONTROLLER.

   This may or may not be the initiator. The list controller has been distinguished from the root for reasons which will become clear later. In short, it may be desirable for someone to have the authority to accept or reject new members, while another party (the root) would actually perform the function.

3. Every participant in the multicast session will be referred to as a GROUP PARTICIPANT. Specific group participants other than the root or list controller will be referred to as LEAVES.

4. After the root checks the security attributes of the participants listed on the multicast group participant list to make sure that they all support the required security attributes, the root will then pass the multicast group list to all other participants and create and distribute the Net Key. If a participant on the multicast group list did not meet the required security attributes, the leaf must be deleted from the list.

   Multiple issues can be raised with the distribution of the multicast group list and Net Key.

a. An issue exists with the time ordering of these functions. The multicast group list could be distributed before or after the link is secured (i.e. the Net Key is distributed).

b. An issue exists when a leaf refuses to join the session. If a leaf refuses to join a session, we can send out a modified list before sending out the Net Key, however sending out modified lists, potentially multiple times, would be inefficient. Instead, the root could continue on, and would not send the Net Key to those participants on the list who rejected the session.

For the scenario architectures which follow, we assume the multicast group list will be distributed to the group participants once before the Net Key is distributed. Unlike the scheme described in [4], we recommend that the multicast group participant list be provided to all leaves. By distributing this list to the leaves, it allows them to determine upfront whether they desire to participate in the multicast group or not, thus saving potentially unnecessary key exchanges.

Four potential key management architectures to distribute keying material for multicast sessions are presented. Recall that the features that are highly desirable for the architecture to possess include the time required to setup the multicast group should be minimized, the number of transmissions should be minimized, and memory/storage requirements should be minimized. As will be seen, the first three proposals each fall short in a different aspect of these desired qualities, whereas the fourth proposal appears to strike a balance in the features desired. Thus, the fourth proposal is the one recommended for general implementation and use.

Please note that these approaches also address securely eliminating users from the multicast group, but don't specifically address adding new users to the multicast group following initial setup because this is viewed as evident as to how it would be performed.

5.1 MANUAL KEY DISTRIBUTION

Through manual key distribution, symmetric key is delivered without the use of public key exchanges. To set up a multicast group Net Key utilizing manual key distribution would require a sequence of events where Net Key and spare Net Keys would be ordered by the root of the multicast session group. Alternate (supersession) Net Keys are ordered (by the root) to be used in case of a compromise of a group participant(s). The Net Keys would be distributed to each individual group participant, often through some centralized physical intermediate location. At some predetermined time, all group participants would switch to the new Net Key. Group participants use this Net Key until a predetermined time when they need another new Net Key. If the Net Key is compromised during this time, the alternate Net Key is used. Group participants switch to the alternate Net Key as soon as they receive it, or upon notification from the root that everyone has the new Net Key and thus the switch over should take place. This procedure is repeated for each cryptoperiod.

A scheme like this may be attractive because the methods exist today and are understood by users. Unfortunately, this type of scheme can be time consuming to set up the multicast group based on time necessary to order keying material and having it delivered. For most real time scenarios, this method is much too slow.

5.2 N Root/Leaf Pairwise Keys Approach

This approach is a brute force method to provide a common multicast group Net Key to the group participants. In this scheme, the initiator sets the security attributes for a particular session, generates a list of desired group participants and transmits the list to all group participants. The leaves then respond with an initial acceptance or rejection of participation. By sending the list up front, time can be saved by not performing key exchanges with people who rejected participation in the session. The root (who for this and future examples is assumed to be the initiator) generates a pairwise key with one of the participants (leaves) in the multicast group using some standard public key exchange technique (e.g., a Diffie-Hellman public key exchange.) The root will then provide the security association parameters of the multicast (which may be different from the parameters of the initial pairwise key) to this first leaf. Parameters may include items such as classification and policy. Some negotiation (through the use of a Security Association Management Protocol, or SAMP) of the parameters may be necessary. The possibility exists for the leaf to reject the connection to the multicast group based on the above parameters and multicast group list. If the leaf rejects this session, the root will repeat this process with another leaf.

Once a leaf accepts participation in the multicast session, these two then choose a Net Key to be used by the multicast group. The Net Key could be generated through another public key exchange between the two entities, or simply chosen by the root, depending upon the policy which is in place for the multicast group (i.e. this policy decision will not be a real time choice). The issue here is the level of trust that the leaf has in the root. If the initial pairwise key exchange provides some level of user authentication, then it seems adequate to just have the root select the Net Key at this stage. Another issue is the level of trust in the strength of the security of the generated key. Through a cooperative process, both entities (leaf and root) will be providing information to be used in the formation of the Net Key.

The root then performs a pairwise key exchange with another leaf and optionally performs the negotiation discussed earlier. Upon acceptance by the leaf to join the multicast group, the root sends the leaf the Net Key.

This pairwise key exchange and Net Key distribution continues for all N users of the multicast group.

Root/leaves cache pairwise keys for future use. These keys serve as Key Encryption Keys (KEKs) used for rekeying leaves in the net at a later time. Only the root will cache all of the leaves' pairwise keys. Each individual leaf will cache only its own unique pairwise Key Encryption Key.

There are two cases to consider when caching the KEKs. The first case is when the Net key and KEK are per session keys. In this case, if one wants to exclude a group participant from the multicast session (and rekey the remaining participants with a new Net Key), the root would distribute a new Net key encrypted with each individual KEK to every legitimate remaining participant. These KEKs are deleted once the multicast session is completed.

The second case to consider is when the KEKs are valid for more than one session. In this case, the Net Key may also be valid for multiple sessions, or the Net Key may still only be valid for one session as in the above case. Whether the Net Key is valid for one session or more than one session, the KEK will be cached. If the Net Key is only valid per session, the KEKs will be used to encrypt new Net Keys for subsequent multicast sessions. The deleting of group participants occurs as in the previous case described above, regardless of whether the Net Key is per session or to be used for multiple sessions.

A scheme like this may be attractive to a user because it is a straightforward extension of certifiable public key exchange techniques. It may also be attractive because it does not involve third parties. Only the participants who are part of the multicast session participate in the keying mechanism. What makes this scheme so undesirable is that it will be transmission intensive as we scale up in numbers, even for the most computationally efficient participants, not to mention those with less capable hardware (tactical, wireless, etc.). Every time the need arises to drop an "unauthorized" participant, a new Net Key must be distributed.

This distribution requires a transmission from the Root to each remaining participant, whereby the new Net Key will be encrypted under the cover of each participant's unique pairwise Key Encryption Key (KEK).

Note: This approach is essentially the same as one proposal to the Internet Engineering Task Force (IETF) Security Subworking Group [Ref 1,2].

Also note that there exist multiple twists to an approach like this. For example, instead of having the root do all N key exchanges, the root could pass some of this functionality (and control) to a number of leaves beneath him. For example, the multicast group list could be split in half and the root tells one leaf to take half of the users and perform a key exchange with them (and then distribute the Net key) while the root will take care of the other half of the list. (The chosen leaves are thus functioning as a root and we can call them "subroots." These subroots will have leaves beneath them, and the subroots will maintain the KEK of each leaf beneath it.) This scales better than original approach as N becomes large. Specifically, it will require less time to set up (or rekey) the multicast net because the singular responsibility of performing pairwise key exchanges and distributing Net Key will be shared among multiple group participants and can be performed in parallel, as opposed to the root only distributing the Net Key to all of the participants.

This scheme is not without its own security concerns. This scheme pushes trust down to each subgroup controller - the root assumes that these "subroot" controllers are acting in a trustworthy way. Every control element (root and subroots) must remain in the system throughout the multicast. This effectively makes removing someone from the net (especially the subroots) harder and slower due to the distributed control. When removing a participant from the multicast group which has functioned on behalf of the root, as a subroot, to distribute Net Key, additional steps will be necessary. A new subroot must be delegated by the root to replace the removed subroot. A key exchange (to generate a new pairwise KEK) must occur between the new subroot and each leaf the removed subroot was responsible for. A new Net Key will now be distributed from the root, to the subroots, and to the leaves. Note that this last step would have been the only step required if the removed party was a leaf with no controlling responsibilities.

5.3 COMPLEMENTARY VARIABLE APPROACH

Let us suppose we have N leaves. The Root performs a public key exchange with each leaf i (i= 1,2, ..., N). The Root will cache each pairwise KEK. Each leaf stores their own KEK. The root would provide the multicast group list of participants and attributes to all users. Participants would accept or reject participation in the multicast session as described in previous sections. The root encrypts the Net Key for the Multicast group to each leaf, using their own unique KEK(i). (The Root either generated this Net Key himself, or cooperatively generated with one of the leaves as was discussed earlier). In addition to the encrypted Net Key, the root will also encrypt something called complementary variables and send them to the leaves.

A leaf will NOT receive his own complementary variable, but he will receive the other N-1 leaf complementary variables. The root sends the Net Key and complementary variables j, where j=1,2,...,N and j not equal to i, encrypted by KEK(i) to each leaf. Thus, every leaf receives and stores N variables which are the Net key, and N-1 complementary variables.

Thus to cut a user from the multicast group and get the remaining participants back up again on a new Net Key would involve the following. Basically, to cut leaf number 20 out of the net, one message is sent out that says "cut leaf 20 from the net." All of the other leaves (and Root) generate a new Net Key based on the current Net Key and Complementary variable 20. [Thus some type of deterministic key variable generation process will be necessary for all participants of the multicast group]. This newly generated variable will be used as the new Net Key by all remaining participants of the multicast group. Everyone except leaf 20 is able to generate the new Net Key, because they have complementary variable 20, but leaf 20 does not.

A scheme like this seems very desirable from the viewpoint of transmission savings since a rekey message encrypted with each individual KEK to every leaf does not have to be sent to delete someone from the net. In other words, there will be one plaintext message to the multicast group versus N encrypted rekey messages. There exists two major drawbacks with this scheme. First are the storage requirements necessary for the (N-1) complementary variables. Secondly, when deleting multiple users from the multicast group, collusion will be a concern. What this means is that these deleted users could work together and share their individual complementary variables to regain access to the multicast session.

5.4 HIERARCHICAL TREE APPROACH

The Hierarchical Tree Approach is our recommended approach to address the multicast key management problem. This approach provides for the following requisite features:

1. Provides for the secure removal of a compromised user from the multicast group

2. Provides for transmission efficiency

3. Provides for storage efficiency

This approach balances the costs of time, storage and number of required message transmissions, using a hierarchical system of auxiliary keys to facilitate distribution of new Net Key. The result is that the storage requirement for each user and the transmissions required for key replacement are both logarithmic in the number of users, with no background transmissions required. This approach is robust against collusion of excluded users. Moreover, while the scheme is hierarchical in nature, no infrastructure is needed beyond a server (e.g., a root), though the presence of such elements could be used to advantage (See Figure 1).

```
                            --------------------------
                           |                          |
                           |        S E R V E R       |
                           |                          |
                            --------------------------
                            |    |                   |
                            |    |     .  .  .  .    |
                            -    -                   -
                           |1|  |2|                 |n|
                            -    -                   -
```

Figure 1: Assumed Communication Architecture

The scheme, advantages and disadvantages are enumerated in more detail below. Consider Figure 2 below. This figure illustrates the logical key distribution architecture, where keys exist only at the server and at the users. Thus, the server in this architecture would hold Keys A through O, and the KEKs of each user. User 11 in this architecture would hold its own unique KEK, and Keys F, K, N, and O.

```
     net key                      Key O
                     -------------------------------------
     intermediate   |                                     |
     keys           |                                     |
                Key M                                 Key N
           -----------------                    --------------------
          |                 |                  |                    |
          |                 |                  |                    |
        Key I             Key J              Key K                Key L
       --------          --------          ---------            ----------
      |        |        |        |        |         |          |          |
      |        |        |        |        |         |          |          |
    Key A    Key B    Key C     Key D    Key E     Key F      Key G      Key H
     ---      ---      ---       ---      ---       ----       ----       ----
    |   |    |   |    |   |     |   |    |   |     |    |     |    |     |    |
    -   -    -   -    -   -     -   -    -   --    --   --    --   --    --   --
   |1| |2|  |3| |4|  |5| |6|   |7| |8|  |9| |10|  |11| |12|  |13| |14|  |15| |16|
    -   -    -   -    -   -     -   -    -   --    --   --    --   --    --   --
                                    users
```

Figure 2: Logical Key Distribution Architecture

We now describe the organization of the key hierarchy and the setup process. It will be clear from the description how to add users after the hierarchy is in place; we will also describe the removal of a user. Note: The passing of the multicast group list and any negotiation protocols is not included in this discussion for simplicity purposes.

We construct a rooted tree (from the bottom up) with one leaf corresponding to each user, as in Figure 2. (Though we have drawn a balanced binary tree for convenience, there is no need for the tree to be either balanced or binary - some preliminary analysis on tree shaping has been performed.) Each user establishes a unique pairwise key with the server. For users with transmission capability, this can be done using the public key exchange protocol. The situation is more complicated for receive-only users; it is easiest to assume these users have pre-placed key.

Once each user has a pairwise key known to the server, the server generates (according to the security policy in place for that session) a key for each remaining node in the tree. The keys themselves should be generated by a robust process. We will also assume users have no information about keys they don't need. (Note: There are no users at these remaining nodes, (i.e., they are logical nodes) and the key for each node need only be generated by the server via secure means.) Starting with those nodes all of whose children are leaves and proceeding towards the root, the server transmits the key for each node, encrypted using the keys for each of that node's children. At the end of the process, each user can determine the keys corresponding to those nodes above her leaf. In particular, all users hold the root key, which serves as the common Net Key for the group. The storage requirement for a user at depth d is d+1 keys (Thus for the example in Figure 2, a user at depth d=4 would hold five keys. That is, the unique Key Encryption Key generated as a result of the pairwise key exchange, three intermediate node keys - each separately encrypted and transmitted, and the common Net Key for the multicast group which is also separately encrypted.)

It is also possible to transmit all of the intermediate node keys and root node key in one message, where the node keys would all be encrypted with the unique pairwise key of the individual leaf. In this manner, only one transmission (of a larger message) is required per user to receive all of the node keys (as compared to d transmissions). It is noted for this method, that the leaf would require some means to determine which key corresponds to which node level.

It is important to note that this approach requires additional processing capabilities at the server where other alternative approaches may not. In the worst case, a server will be responsible for generating the intermediate keys required in the architecture.

5.4.1 The Exclusion Principle

Suppose that User 11 (marked on Figure 2 in black) needs to be deleted from the multicast group. Then all of the keys held by User 11 (bolded Keys F, K, N, O) must be changed and distributed to the users who need them, without permitting User 11 or anyone else from obtaining them. To do this, we must replace the bolded keys held by User 11, proceeding from the bottom up. The server chooses a new key for the lowest node, then transmits it encrypted with the appropriate daughter keys (These transmissions are represented by the dotted lines). Thus for this example, the first key replaced is Key F, and this new key will be sent encrypted with User 12's unique pairwise key.

Since we are proceeding from the bottom up, each of the replacement keys will have been replaced before it is used to encrypt another key. (Thus, for the replacement of Key K, this new key will be sent encrypted in the newly replaced Key F (for User 12) and will also be sent as one multicast transmission encrypted in the node key shared by Users 9 and 10 (Key E). For the replacement of Key N, this new key will be sent encrypted in the newly replaced Key K (for Users 9, 10, and 12) and will also be encrypted in the node key shared by Users 13, 14, 15, and 16 (Key L). For the replacement of Key O, this new key will be sent encrypted in the newly replaced Key N (for Users 9, 10, 12, 13, 14, 15, and 16) and will also be encrypted in the node key shared by Users 1, 2 , 3, 4, 5, 6, 7, and 8 (Key M).) The number of transmissions required is the sum of the degrees of the replaced nodes. In a k-ary tree in which a sits at depth d, this comes to at most kd-1 transmissions. Thus in this example, seven transmissions will be required to exclude User 11 from the multicast group and to get the other 15 users back onto a new multicast group Net Key that User 11 does not have access to. It is easy to see that the system is robust against collusion, in that no set of users together can read any message unless one of them could have read it individually.

If the same strategy is taken as in the previous section to send multiple keys in one message, the number of transmissions required can be reduced even further to four transmissions. Note once again that the messages will be larger in the number of bits being transmitted. Additionally, there must exist a means for each leaf to determine which key in the message corresponds to which node of the hierarchy. Thus, in this example, for the replacement of keys F, K, N, and O to User 12, the four keys will be encrypted in one message under User 12's unique pairwise key. To replace keys K, N, and O for Users 9 and 10, the three keys will be encrypted in one message under the node key shared by Users 9 and 10 (Key E). To replace keys N and O for Users 13, 14, 15, 16, the two keys will be encrypted in one message under the node key shared by Users 13, 14, 15, and 16 (Key L). Finally, to replace key O for Users 1, 2 , 3, 4, 5, 6, 7, and 8, key O will be encrypted under the node key shared by Users 1, 2 , 3, 4, 5, 6, 7, and 8 (Key M). Thus the number of transmission required is at most (k-1)d.

The following table demonstrates the removal of a user, and how the storage and transmission requirements grow with the number of users.

Table 1: Storage and Transmission Costs

Number of users	Degree (k)	Storage per user (d+1)	Transmissions to rekey remaining participants of multicast group- one key per message (kd-1)	Transmissions to rekey remaining participants of multicast group - multiple keys per message (k-1)d
8	2	4	5	3
9	3	3	5	4
16	2	5	7	4
2048	2	12	21	11
2187	3	8	20	14
131072	2	18	33	17
177147	3	12	32	22

The benefits of a scheme such as this are:

1. The costs of user storage and rekey transmissions are balanced and scalable as the number of users increases. This is not the case for [1], [2], or [4].

2. The auxiliary keys can be used to transmit not only other keys, but also messages. Thus the hierarchy can be designed to place subgroups that wish to communicate securely (i.e. without transmitting to the rest of the large multicast group) under particular nodes, eliminating the need for maintenance of separate Net Keys for these subgroups. This works best if the users operate in a hierarchy to begin with (e.g., military operations), which can be reflected by the key hierarchy.

3. The hierarchy can be designed to reflect network architecture, increasing efficiency (each user receives fewer irrelevant messages). Also, server responsibilities can be divided up among subroots (all of which must be secure).

4. The security risk associated with receive-only users can be minimized by collecting such users in a particular area of the tree.

5. This approach is resistant to collusion among arbitrarily many users.

As noted earlier, in the rekeying process after one user is compromised, in the case of one key per message, each replaced key must be decrypted successfully before the next key can be replaced (unless users can cache the rekey messages). This bottleneck could be a problem on a noisy or slow network. (If multiple users are being removed, this can be parallelized, so the expected time to rekey is roughly independent of the number of users removed.)

By increasing the valences and decreasing the depth of the tree, one can reduce the storage requirements for users at the price of increased transmissions. For example, in the one key per message case, if n users are arranged in a k-ary tree, each user will need storage. Rekeying after one user is removed now requires transmissions. As k approaches n, this approaches the pairwise key scheme described earlier in the paper.

5.4.2 Hierarchical Tree Approach Options

5.4.2.1 Distributed Hierarchical Tree Approach

The Hierarchical Tree Approach outlined in this section could be distributed as indicated in Section 5.2 to more closely resemble the proposal put forth in [4]. Subroots could exist at each of the nodes to handle any joining or rekeying that is necessary for any of the subordinate users. This could be particularly attractive to users which do not have a direct connection back to the Root. Recall as indicated in Section 5.2, that the trust placed in these subroots to act with the authority and security of a Root, is a potentially dangerous proposition. This thought is also echoed in [4].

Some practical recommendations that might be made for these subroots include the following. The subroots should not be allowed to change the multicast group participant list that has been provided to them from the Root. One method to accomplish this, would be for the Root to sign the list before providing it to the subroots. Authorized subroots could though be allowed to set up new multicast groups for users below them in the hierarchy.

It is important to note that although this distribution may appear to provide some benefits with respect to the time required to initialize the multicast group (as compared to the time required to initialize the group as described in Section 5.4) and for periodic rekeying, it does not appear to provide any benefit in rekeying the multicast group when a user has been compromised.

It is also noted that whatever the key management scheme is (hierarchical tree, distributed hierarchical tree, core based tree, GKMP, etc.), there will be a "hit" incurred to initialize the multicast group with the first multicast group net key. Thus, the hierarchical tree approach does not suffer from additional complexity with comparison to the other schemes with respect to initialization.

5.4.2.2 Multicast Group Formation

Although this paper has presented the formation of the multicast group as being Root initiated, the hierarchical approach is consistent with user initiated joining. User initiated joining is the method of multicast group formation presented in [4]. User initiated joining may be desirable when some core subset of users in the multicast group need to be brought up on-line and communicating more quickly. Other participants in the multicast group can then be brought in when they wish. In this type of approach though, there does not exist a finite period of time by when it can be ensured all participants will be a part of the multicast group.

For example, in the case of a single root, the hierarchy is set up once, in the beginnning, by the initiator (also usually the root) who also generates the group participant list. The group of keys for each participant can then be individually requested (pulled) as soon as, but not until, each participant wishes to join the session.

5.4.2.3 Sender Specific Authentication

In the multicast environment, the possibility exists that participants of the group at times may want to uniquely identify which participant is the sender of a multicast group message. In the multicast key distribution system described by Ballardie [4], the notion of "sender specific keys" is presented.

Another option to allow participants of a multicast group to uniquely determine the sender of a message is through the use of a signature process. When a member of the multicast group signs a message with their own private signature key, the recipients of that signed message in the multicast group can use the sender's public verification key to determine if indeed the message is from who it is claimed to be from.

Another related idea to this is the case when two users of a multicast group want to communicate strictly with each other, and want no one else to listen in on the communication. If this communication relationship is known when the multicast group is originally set up, then these two participants could simply be placed adjacent to one another at the lowest level of the hierarchy (below a binary node). Thus, they would naturally share a secret pairwise key. Otherwise, a simple way to accomplish this is to perform a public key based pairwise key exchange between the two users to generate a traffic encryption key for their private unicast communications. Through this process, not only will the encrypted transmissions between them be readable only by them, but unique sender authentication can be accomplished via the public key based pairwise exchange.

5.4.2.4 Rekeying the Multicast Group and the Use of Group Key Encryption Keys

Reference [4] makes use of a Group Key Encryption Key that can be shared by the multicast group for use in periodic rekeys of the multicast group. Aside from the potential security drawbacks of implementing a shared key for encrypting future keys, the use of a Group Key Encryption Key is of no benefit to a multicast group if a rekey is necessary due to the known compromise of one of the members. The strategy for rekeying the multicast group presented in Section 5.4.1 specifically addresses this critical problem and offers a means to accomplish this task with minimal message transmissions and storage requirements.

The question though can now be asked as to whether the rekey of a multicast group will be necessary in a non-compromise scenario. For example, if a user decides they do not want to participate in the group any longer, and requests the list controller to remove them from the multicast group participant list, will a rekey of the multicast group be necessary? If the security policy of the multicast group mandates that deleted users can no longer receive transmissions, than a rekey of a new net key will be required. If the multicast group security policy does not care that the deleted person can still decrypt any transmissions (encrypted in the group net key that they might still hold), but does care that they can not encrypt and transmit messages, a rekey will once again be necessary. The only alternative to rekeying the multicast group under this scenario would require a recipient to check every received message sender, against the group participant list. Thus rejecting any message sent by a user not on the list. This is not a practical option. Thus it is recommended to always rekey the multicast group when someone is deleted, whether it is because of compromise reasons or not.

5.4.2.5 Bulk Removal of Participants

As indicated in Section 2, the need may arise to remove users in bulk. If the users are setup as discussed in Section 5.4.1 into subgroups that wish to communicate securely all being under the same node, bulk user removal can be done quite simply if the whole node is to be removed. The same technique as described in Section 5.4.1 is performed to rekey any shared node key that the remaining participants hold in common with the removed node.

The problem of bulk removal becomes more difficult when the participants to be removed are dispersed throughout the tree. Depending on how many participants are to be removed, and where they are located within the hierarchy, the number of transmissions required to rekey the multicast group could be equivalent to brute force rekeying of the remaining participants. Also the question can be raised as to at what point the remaining users are restructured into a new hierarchical tree, or should a new multicast group be formed. Restructuring of the hierarchical tree would most likely be the preferred option, because it would not necessitate the need to perform pairwise key exchanges again to form the new user unique KEKs.

5.4.2.6 ISAKMP Compatibility

Thus far this document has had a major focus on the architectural trade-offs involved in the generation, distribution, and maintenance of traffic encryption keys (Net Keys) for multicast groups. There are other elements involved in the establishment of a secure connection among the multicast participants that have not been discussed in any detail. For example, the concept of being able to "pick and choose" and negotiating the capabilities of the key exchange mechanism and various other elements is a very important and necessary aspect.

The NSA proposal to the Internet Engineering Task Force (IETF) Security Subworking Group [Ref. 3] entitled "Internet Security Association and Key Management Protocol (ISAKMP)" has attempted to identify the various functional elements required for the establishment of a secure connection for the largest current network, the Internet. While the proposal has currently focused on the problem of point to point connections, the functional elements should be the same for multicast connections, with appropriate changes to the techniques chosen to implement the individual functional elements. Thus the implementation of ISAKMP is compatible with the use of the hierarchical tree approach.

6.0 SUMMARY

As discussed in this report, there are two main areas of concern when addressing solutions for the multicast key management problem. They are the secure initialization and rekeying of the multicast group with a common net key. At the present time, there are multiple papers which address the initialization of a multicast group, but they do not adequately address how to efficiently and securely remove a compromised user from the multicast group.

This paper proposed a hierarchical tree approach to meet this difficult problem. It is robust against collusion, while at the same time, balancing the number of transmissions required and storage required to rekey the multicast group in a time of compromise.

It is also important to note that the proposal recommended in this paper is consistent with other multicast key management solutions [4], and allows for multiple options for its implementation.

7.0 Security Considerations

Security concerns are discussed throughout this memo.

8.0 REFERENCES

1. Harney, H., Muckenhirn, C. and T. Rivers, "Group Key Management Protocol Architecture", RFC 2094, September 1994.

2. Harney, H., Muckenhirn, C. and T. Rivers, "Group Key Management Protocol Specification", RFC 2093, September 1994.

3. Maughan, D., Schertler, M. Schneider, M. and J.Turner, "Internet Security Association and Key Management Protocol, Version 7", February 1997.

4. Ballardie, T., "Scalable Multicast Key Distribution", RFC 1949, May 1996.

5. Wong, C., Gouda, M. and S. Lam, "Secure Group Communications Using Key Graphs", Technical Report TR 97-23, Department of Computer Sciences, The University of Texas at Austin, July 1997.

Authors' Addresses

Debby M. Wallner
National Security Agency
Attn: R2
9800 Savage Road STE 6451
Ft. Meade, MD. 20755-6451

Phone: 301-688-0331
EMail: dmwalln@orion.ncsc.mil

Eric J. Harder
National Security Agency
Attn: R2
9800 Savage Road STE 6451
Ft. Meade, MD. 20755-6451

Phone: 301-688-0850
EMail: ejh@tycho.ncsc.mil

Ryan C. Agee
National Security Agency
Attn: R2
9800 Savage Road STE 6451
Ft. Meade, MD. 20755-6451

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.