KEY MANAGEMENT FOR SECURE MULTICAST IN HYBRID SATELLITE NETWORKS

Ayan Roy-Chowdhury and John S. Baras
Department of Electrical and Computer Engineering and
Institute for Systems Research
University of Maryland College Park, MD 20742,USA
ayan@umd.edu; baras@isr.umd.edu

Abstract This paper proposes a design for key management for secure multicast in hybrid satellite networks. Communication satellites offer an efficient way to extend IP multicast services for groups in wide-area networks. In order to be commercially viable, the multicast traffic should be accessible only to paying subscribers. Access control can be achieved by data encryption. This requires secure and efficient methods to generate, distribute and update the keys. Most current key management protocols do not scale well when applied to large dynamic groups in wide-area networks. This paper attempts to solve the above problem for groups in a hybrid network that is composed of terrestrial Ethernet LANs interconnected by ATM-based satellite channels. We investigate current group key management protocols, and design a framework for secure and scalable key management for the multicast routing architecture in the satellite network. The proposed framework is presented in detail, alongwith analysis and simulation results.

Keywords: Satellite network, secure multicast, group key management.

1. INTRODUCTION

Multicasting is a network-layer mechanism for one-to-many or many-to-many communication that is efficient in terms of usage of network resources. With the growth of the Internet, web applications using Internet Protocol (IP) based multicast routing protocols are becoming increasingly popular. Examples are webcasts, video and voice conferencing and Internet gaming.

Satellite networks offer a natural method to extend the multicast services in wide-area networks where the sources and recipients are widely separated from one another. There is, however, little support today for IP multicast services over satellites. Apart from the problems involved in creating an efficient routing mechanism for satellite multicast, another major challenge is to secure the multicast data in the satellite network. The IP multicast paradigm allows free

Y. Deswarte et al. (eds.), *Security and Protection in Information Processing Systems*
© Springer Science+Business Media Dordrecht 2004

access to the multicast data to anyone interested in receiving it. However, in order for a multicast service to be commercially viable, access to the multicast data should be restricted to paying or authorized receivers. Access control can be achieved by means of encryption - the source encrypts the application content using a key; the decryption key is distributed to all authorized receivers. The mechanism of key distribution is challenging when the set of authorized receivers changes dynamically with time. The design problem becomes more complex when we consider large groups of the order of thousands of members, spread over a wide geographical area, as might be the case for satellite networks.

In [Roy-Chowdhury, 2003] we have proposed a multicast routing architecture that scales to large groups in a wide-area hybrid satellite network. In this paper we address the problem of group key management for secure multicast in the above network. We propose a framework for secure key management for groups operating in this network, with the primary objective of minimizing the communication over the satellite links, and present simulation results to demonstrate the feasibility of the proposed framework.

The rest of the paper is organized as follows. We review current proposals for group key management in section 2. Section 3 describes in brief the network architecture. The design of the key management framework is given in section 4. Various analyses of the framework are in sections 5, 6. We describe our simulation and results in section 7. We conclude in section 8, highlighting future research directions.

2. REVIEW OF KEY MANAGEMENT PROTOCOLS

We describe in brief some of the fundamental ideas presented in group key management. A more detailed analysis of the protocols presented here can be found in [Roy-Chowdhury, 2003].

Most of the protocols proposed to date fall in two categories: centralized key distribution schemes and distributed key generation schemes. In centralized key distribution, there is a centralized key controller to whom all members send join and leave requests. The key controller is fully trusted and is responsible for key generation and distribution to the group members, and for key updates, triggered periodically or on membership changes. The centralized schemes provide a higher degree of security and are more efficient. Their major weakness is the dependence on a central entity, which can be a single point of failure. The Key Predistribution System, proposed in [Matsumoto and Imai, 1988], and the Broadcast Encryption scheme proposed in [Fiat and Naor, 1994] are examples in which the key controller pre-computes the group keys for all possible groups. In these schemes, the memory requirements can become prohibitively high for large groups. Another category of centralized schemes are

the *threshold encryption* protocols, such as [Berkovits, 1991]. They require collaboration between participants (who might not know each other, as in IP multicast), and might have high storage requirements for large groups. Secure Lock is a secure broadcasting scheme proposed in [Chiou and Chen, 1989]. Here the number of key encryptions done at the centralized controller increases linearly with the number of group members. The system is one-to-many, and cannot be used if there are multiple sources. Another one-to-many system is the Conditional Access System (CAS) [Kim et al., 1996], which is popular for data confidentiality in satellite broadcasts. Group Key Management Protocol (GKMP) [Harney and Muckenhirn, 1997] has been proposed for groups with multiple sources and receivers. In GKMP, the communication overhead in sending messages for the initial system setup, and key update messages on member leaves, is high for large groups.

In distributed key generation schemes all the group members (or a chosen subset), contribute shares in a round of message exchanges to generate a common group key. Key agreement for secure multicast using hidden fractional keys (HFK), proposed in [Poovendran, 1999], is an example. The scheme does not handle membership changes well. A suite of protocols have been proposed in [Steiner et al., 2000] for fully distributed group key agreement. In this system, the computational burden on each entity for generating the group key, can be prohibitively high for large groups. Both the preceding protocols, and distributed schemes in general, require that all members participating in the key setup are aware of one another, and can send messages in order to the others. This is not necessarily true in several group situations, such as IP multicast. Also, the protocols in general suffer from a high overhead in communication for key generation when applied to large groups. Several other distributed protocols have been proposed in [Burmester and Desmedt, 1994; Steer et al., 1990]; all are susceptible to similar inefficiency problems in large groups.

A family of protocols have been proposed for key management based on logical trees, originally in [Wong et al., 2000; Wallner et al., 1999]. The original protocol is called the Logical Key Hierarchy (LKH). The centralized logical tree-based protocols have a group controller (GC), who constructs a logical key tree with the group members at the leaf nodes of the tree. The internal nodes of the tree are usually logical nodes and correspond to the key encrypting keys (KEK) which are used to securely transport key updates to the group. The root of the tree is the session key or traffic encrypting key (TEK). The key tree protocols have logarithmic communication, storage and computation complexity. These protocols scale very well to large groups. Their primary drawback is the use of a centralized GC; [Rodeh et al., 2000] has suggested a distributed version that does not have a GC. Various modifications to the original LKH protocol have been made that try to reduce the communication

and computational complexity, for example, [Canetti et al., 1999; Perrig et al., 2001].

3. NETWORK ARCHITECTURE

The network architecture is given in figure 1. We consider a group of terrestrial Ethernet-based networks geographically separated and spread over a wide area. We term them the "subnetworks" in our overall network. Each subnetwork has one or more satellite gateways, which interconnect the subnetworks via ATM-based satellite links using a geostationary satellite.

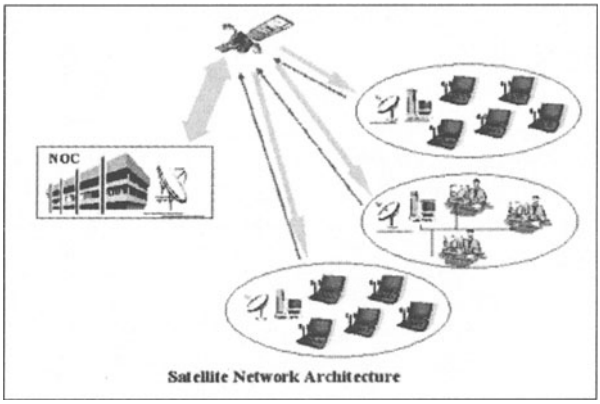

Figure 1. The Network Architecture

In [Roy-Chowdhury, 2003], we have described a routing architecture whereby sources and receivers spread across different subnetworks can form an IP multicast group. The IP multicast framework has two components: the multicast routing within a subnetwork is based on *Protocol Independent Multicast - Sparse Mode* [Deering et al., 1996], while the multicast routing between the subnetworks over the ATM satellite links uses the ATM *Multicast Address Resolution Server* (MARS) with VC mesh architecture [Armitage, 1997]. A satellite gateway router in each subnetwork acts as the root of the multicast tree within its subnetwork. This router is known as the Rendezvous Point (RP). The MARS is used for address mapping for IP multicast over the ATM links. It is located at the Network Operations Center (NOC). The key management framework proposed in this paper builds on the multicast routing architecture.

4. TIERED KEY MANAGEMENT IN SATELLITE ATM NETWORK

The primary metric that we consider for our design is the communication overhead in the network. The propagation delay in the geostationary satellite

links is high, of the order of 250ms in one hop. The uplink bandwidth is limited to 1.5Mbps. Also, geostationary satellites operating in the Ka-band are prone to channel errors due to atmospheric conditions such as rain fade. We therefore need a key management scheme that minimizes the communication over the satellite links, to reduce the delay in group initialization or key updates, and also to minimize the possibility of error conditions where the group keys do not reach all the members due to channel conditions. We assume that the hosts in each terrestrial network have significant processing power and memory capacity, similar to the workstations and personal computers prevalent today. Hence computation and storage are not critical issues.

The hierarchical structure of the network creates two distinct levels in the network - the terrestrial subnetworks, and the satellite connections between the subnetworks forming an "overlay", as shown in figure 2. We consider the different subnetworks to be independent domains, such as company networks, which might follow different security policies. Reconciling the security policies across the subnetworks to build a single key management framework would be a difficult task. Also, a single key framework would suffer from the *1-affects-n* scalability problem [Mittra, 1997] and the probability of updates in the keys stored at a member would be much higher due to the dynamics of member joins and leaves overall. Join or leave of a member in any subnetwork would trigger updates even in remote subnetworks, which is highly inefficient. The key management communication over the satellite links would hence be frequent, which is undesirable from our perspective due to the reasons mentioned above.

We therefore divide the key management into two tiers - one at the subnetwork level, while the other at the level of the satellite overlay (the concept of dividing a system into subgroups for scalable key management was originally proposed in Iolus [Mittra, 1997]). The key generation and distribution in each subnetwork is independent of one another, and also of the key generation and distribution in the overlay; we add mechanisms so that the encrypted data can be transferred securely across the different key management areas. The key management in each logical group is based on centralized key trees, which is selected due to its scalability to large groups, in terms of the communication required to initialize and update the group keys. The framework therefore has two tree levels: a global *RP Tree* for managing the keys between the subnet RPs in the overlay; and the local *SN Tree* for managing the keys amongst the hosts in each subnet. Each subnetwork has its own SN Tree. We term this framework, *Tiered Tree Based Key Management.*

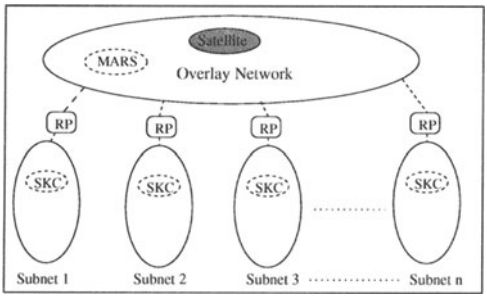

Figure 2. Logical Grouping in the Satellite Network

4.1 Trust Model and Security Assumptions

The network entities that are relevant in the security framework are the ATM multicast server (MARS), the Rendezvous Points and Key Server in each subnetwork and the end-hosts.

In the routing framework, the MARS maintains the database of multicast group membership at the subnetwork level. It periodically sends the group membership information to all the RPs that are subscribed to the group. We envision the MARS, located at the NOC, to be owned by the network service provider, and it keeps track of the different customers (which can be each subnetwork) who are using the network services provided. The customers would prefer to keep their traffic confidential and not allow the network provider read the transmissions. Therefore in the security framework, we model the MARS as the trusted third party for performing access control, based on group policy, for different subnetworks that want to join or send to a given multicast group. In addition, the MARS acts as a Certificate Authority (CA) for verifying the public keys of the RPs when needed. However, the MARS is not trusted with the multicast traffic. The MARS should not receive the application data (unless it explicitly subscribes as a member to the multicast group).

The RP in each subnetwork is located at the satellite terminal, which is the communication gateway to/from all entities outside the subnetwork. In the security design, the RP is trusted to securely transmit multicast data from local sources to remote subnetworks over the satellite links, and to receive multicast data from remote sources in case there are local receivers. However, the RP is not trusted to read the multicast traffic. We place this limitation since the satellite gateway in each subnetwork would usually be owned by the network provider, who might not be authorized to read group information as discussed previously.

The end-hosts are trusted to securely encrypt or decrypt the multicast traffic.

Since we choose centralized key management schemes, we need a group controller/key server for the hosts in a subnetwork. The RP would have been

a good candidate for the key server since it is the root of the multicast tree in its subnetwork. However, as stated above, it might be undesirable to allow the RP to read the group traffic, which it can easily do if it generates the encryption/decryption keys for the hosts. Therefore the security framework introduces a key server in each subnetwork, distinct from the RP, and responsible for managing group keys in its subnet. It is termed the *Subnetwork Key Controller* (SKC). The SKC does access control operations on local group members, and performs key generation, distribution and periodic key updates for all groups that have members in its local subnet. Each end-host and the RP is assumed to *apriori* establish a *secure channel* to the SKC for receiving the key information.

In addition to the above, we make the assumption that the IP/ATM multicast routing is secure.

4.2 Key Management in the Overlay: RP Tree

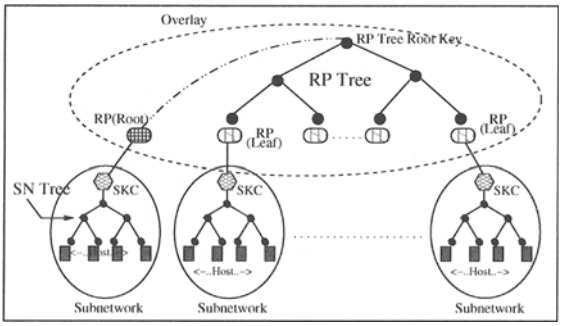

Figure 3. RP Tree and SN Tree

Figure 3 illustrates the key trees for the overlay and each subnetwork in our framework. The logical key tree in the overlay is the RP Tree, that in each subnetwork is the SN Tree. The RPs in different subnetworks are located at the leaves of the RP tree. The root of the RP tree for any group is one of the RPs in the group, while the intermediate nodes as shown in figure 3 are logical nodes which correspond to the KEKs in the key tree. Likewise, in the SN Tree, the SKC is the root, while the hosts are at the leaf nodes of the tree. The RP Tree is described below, and the SN Tree is described in the next section.

RP Tree Setup

The RP tree is constructed by making additions to the MARS message exchange protocol, which is described in [Armitage, 1997].

Sender RP Request: When a RP has hosts downstream who want to send to group G, the RP sends a request message to the MARS for the list of group members. If the MARS database has a non-empty entry of RPs subscribed to

G, the MARS adds the requesting RP to the entry, and returns the details of the entry to the requesting RP in a reply message. The reply message is broadcast to all RPs in G present in the MARS entry at that time. The message has the IP address and public key of each valid RP, and the address of the RP Tree root. If MARS has no entry for G (i.e., the requesting RP is the first to join G at the MARS), then MARS creates a new entry for G, adds the requesting RP ATM address to the entry, and sends a negative acknowledgment in reply.

Receiver RP Join: When a RP has hosts in its local subnetwork requesting to join a group G as receivers, the RP sends a join request to the MARS. The MARS adds the joining RP's address, public key to the database entry at MARS for group G. If the entry does not exist, then a new entry is created. Subsequently the MARS broadcasts the list of RP group members in a regular membership update message to all the RPs subscribed to G. The public key of the RPs are needed to bootstrap the RP Tree, since we do not assume that the different RPs have secure associations between one another established previously. Once the multicast tree and RP key tree state is created in local memory, for subsequent join or send requests from downstream nodes, an RP does not send MARS requests.

Selection of the RP Tree Root: The root of the RP tree is selected to be the sender RP that is the *earliest to join* the group amongst the sender RPs in the MARS database entry. The selection is done by the MARS based on the join time in the requests it receives. The address and public key information of the root RP becomes known to all the group RPs from the MARS message they receive. In case the root RP leaves the group, the MARS checks the joining times of the remaining sender RPs, selects the earliest-to-join, and broadcasts a new message to the group. The RPs update their local group security information upon receiving the MARS message.

Tree Setup at the Root: When a sender RPs receives the MARS message, it checks whether it is the root. If so, it proceeds to set up the logical key tree in its local node. The information about the leaves of the key tree are obtained from the MARS message.

Key Information Transmission: Once the RP tree has been setup at the root, the root creates one message containing all the keys of the RP tree, encrypted as appropriate, and broadcasts the message over the satellite links to all the other RPs in the group. We assume that the root RP has no prior secure association with the leaf RPs. So when the RP tree is created, the initial communication from the root to the leaf RPs are encrypted with the public keys of the leaf RPs. Upon reception, each leaf RP decrypts its relevant key information using its private key, and obtains all the keys on the path from its leaf to the root of the tree. The key corresponding to the tree root is now used as the session key. Subsequent communication from the root RP to the leaf RPs uses

the shared long-term secret, corresponding to the leaf node of each receiver RP, that is sent by the root in the initial communication.

RP Tree Update on Member Join, Leave

When a RP wants to join an existing group as a member, it sends a join request to the MARS. The MARS adds the RP to the group entry. When a *leaf* RP leaves a group it sends a leave request to the MARS for the group. The MARS ensures that the leaving RP is not the RP tree root and removes the RP information from the group entry. The join or leave message is retransmitted to the existing group members to update them about change in the group membership. On getting the join/leave message, the root RP updates the keys in the RP tree as required, and sends the updated keys to the affected group members. When the root RP sends a leave message, the MARS removes the root from the group entry; runs the algorithm to select a new root RP; creates a new update message and immediately sends the update to the remaining group members. The new root, upon receiving the update message, proceeds to create a new RP tree. Till the new tree is created, the group information is secured using the existing session key. The drawback is that the old root RP can still receive all the information, but it prevents "blackout periods".

In case the multicast group has only one sender RP (the root) (in situations where there is only one source host, or all the sources are concentrated in the same subnet), the root RP leaving implies there are no sources left. The MARS on getting the leave message cannot locate a new root and hence does not send out a new update message. The group entry will be erased from the MARS database on a timeout.

4.3 Key Management in the Subnetwork: SN Tree

The key server in each subnet, known as the Subnetwork Key Controller (SKC), manages the subnetwork key tree (SN tree). We assume that the security module in all hosts and the RP are aware of the address of the SKC.

SN Tree Setup: When an end-host wants to join a multicast group G as a receiver, or intends to send to a multicast group as a sender, it first sends a *join request* message to the SKC specifying the IP address of G. In the subnet, the SKC does not differentiate between a sending host and a receiving host.

When the SKC receives a join request, it checks its local database for an entry for the group. If none exists, the SKC creates an entry and the corresponding key tree. The SKC also generates a *datahiding key* for the group. The datahiding key for group G has to be identical across subnetworks; the SKC in a subnetwork has to contact the SKCs in other subnetworks (that have members in G) to agree on the datahiding key for G. The datahiding key is long-term; once created, it does not change for the lifetime of group G, despite member joins and leaves. The SKC assigns the joining host to a leaf in the tree.

It then encrypts all the keys in the path from the leaf node to the root and the datahiding key using the long-term secret it shares with the joining host; it also encrypts only the session key for the RP. The SKC then forms a *key information* message containing the encrypted keys, and transmits the key information message to the host and the local RP. The host decrypts the tree keys and group datahiding key and stores them in local memory. The RP decrypts the session key, creates an entry for the group in local memory, and stores the session key in the entry.

When there are existing group members, or multiple members joining simultaneously, the message will contain all the relevant tree keys encrypted for all affected members.

SN Tree Update on Member Join: When one host sends a join request for group G to the SKC, the controller adds the host to the key tree following the standard procedure for adding group members in LKH, and sends the updated group keys to all the members. The local RP is also informed about the update in the session key. The new member gets all the keys in the path from its root to the leaf in the SN Tree, and also the datahiding key. For multiple members joining simultaneously, the sequence is similar, with the added processing at the SKC to find the minimum number of valid KEKs to send the update information.

SN Tree Update on Member Leave: When a member leaves, all the keys on the path from the member leaf to the root are invalidated. The SKC generates new keys in replacement, and sends the fresh keys to all affected members, and the RP. For bulk member revocation, the SKC has to identify all the invalid keys, and find the minimal number of valid keys that are required to transmit the updated keys.

Synchronization of Group Information at the RP: At all times, the RP maintains integrated state information for a group. When the RP is a leaf of the RP tree, the group entry in its local memory specifies it is a leaf, and contains the path keys to the root of the RP tree, and also the local subnetwork session key. If a leaf RP becomes a root (in situations where the previous root RP has left the group), then a root entry is created. The subnetwork session key is transferred from the leaf entry to the root entry. Note however, a root RP for group G does not become a leaf RP for G at any time when it is continuously subscribed to G.

4.4 Secure Data Transmission in a Group

Multicast traffic can be transmitted securely when the SN trees and the RP tree have been established. The sequence is described here.

1 Source host j in subnetwork i, a_{ij}, encrypts the data m for group G twice: first using the datahiding key DK_G to produce ciphertext $C =$

$E_{DK_G}(m)$. The encrypted data is re-encrypted using the subnetwork session key SK_{G_i} to produce ciphertext $\hat{C} = E_{SK_{G_i}}(C)$.

2 a_{ij} sends the doubly-encrypted data to the local multicast tree and the RP.

3 The group members a_{ik} in the local multicast tree decrypt $\hat{C}$ to retrieve the multicast traffic: $C = D_{SK_{G_i}}\left(\hat{C}\right)$, $m = D_{DK_G}(C)$.

4 The RP decrypts $\hat{C}$ to obtain C. It cannot decrypt C to get m, since it does not know DK_G. The RP re-encrypts C with the RP tree session key $SK_{G_{RP}}$ and transmits the ciphertext $\hat{C}' = E_{SK_{G_{RP}}}(C)$ to the other subnetworks over the satellite link.

5 RP_j in subnetwork j receives the encrypted transmission. It decrypts $\hat{C}'$ to obtain $C = D_{SK_{G_{RP}}}\left(\hat{C}'\right)$. RP_j cannot decrypt C since it does not know DK_G. It re-encrypts C using the local subnetwork session key SK_{G_j} for G to generate ciphertext $\hat{C}'' = E_{SK_{G_j}}(C)$; RP_j sends $\hat{C}''$ along the multicast tree in its subnet.

6 Each host a_{jk} in subnetwork j subscribed to G receives $\hat{C}''$. It decrypts the ciphertext using SK_{G_j} to obtain C. a_{jk} decrypts C using the datahiding key DK_G to obtain m: $m = D_{DK_G}(C)$.

5. SECURITY ANALYSIS

5.1 Passive Adversary

SN Tree: Let A be a passive adversary, who is never a group member. We assume A eavesdrops on all traffic in an arbitrary subnetwork and receives all the encrypted key information and data packets. A cannot decrypt the data packets, since it does not know either the subnetwork session key or the datahiding key. A brute-force attack to find the group key takes $\Omega\left(2^k\right)$ operations where k is the length of the group key. A cannot do better than this, since it does not know any of the KEKs in the tree.

RP Tree: We assume A has the capability of listening to the satellite traffic and receives all the traffic in a complete session, that is, A can be a passive eavesdropping RP. A still cannot decrypt the encrypted traffic, since it does not know the RP session key. It cannot obtain the session key from the RP tree key messages, because it does not have any of the keys used to decrypt the key messages.

MARS: If the MARS is a passive adversary, then under normal operation of the network, the multicast traffic will not reach it at all, since the routing path from a source RP to the set of receiver RPs will not include the MARS.

5.2 Active Adversary

SN Tree: Let B be an active adversary, who has been a group member during some previous time period. In the key management protocol, when B joins the group in any subnet, it cannot derive any previous group key by doing better than exhaustive search, i.e., $\Omega\left(2^k\right)$ operations. Even if B has listened to and stored past group traffic, it cannot obtain any of the decryption keys for the previous enciphered messages. The only keys it gets are the updated keys that are sent to it by the SKC.

Assume B leaves the group and tries to read the group traffic after it has left. B has with it the set of keys on its key path, and the datahiding key. However, it cannot read the group traffic at a later time, since the key server updates all the keys on the key path that B knows, including the session key, and securely transmits the updated keys to the other members using long-term keys that B does not know. The datahiding key does not change. But this does not help B since it first needs to decrypt using the current session key, which B does not possess.

RP Tree: Let B be an RP who was a member of the group at some previous time. Before B had joined the RP tree, it could not decrypt the data traffic since it did not know the group key at a previous time instant. After B joins the RP tree and collects the keys in its key path, it leaves. Once B has left, the root of the tree (assuming B was not the root), updates all the keys in the RP tree known to B, including the RP session key. B cannot obtain the updated keys from the key message since it does not know the decryption keys used to send the updated keys to the other RPs.

The only time when B, as an RP, could read the data after leaving, is if B was the immediate previous root of the RP tree. Then for the interval of time it takes the new root to create a new tree, the group traffic would continue to be encrypted using the old RP Tree session key, allowing B access to the data. However, B can obtain only the ciphertext of the data, encrypted with the datahiding key, which B does not know.

MARS: The MARS can join a multicast group by adding its ATM address to the list of addresses for the multicast group, and sending the list to the source RPs. The routing paths created by the source RPs will then include a branch to the MARS. Subsequently the MARS will receive all the key traffic on the RP tree, and all the encrypted multicast traffic. But even in this situation, the MARS will not be able to read the multicast data, because the multicast traffic is first encrypted with the datahiding key, to which the MARS does not have access.

	RP Root	SKC
Tree setup	$(n_1 - 1)\,k_p + \frac{d_1(n_1-1)}{d_1-1} k_s$	$\left(n_2 + \frac{d_2(n_2-1)}{d_2-1} + 1\right) k_s$
Member join to existing group in subnet	0	$(d_2 h_2 + 1)\,k_s + k_s$
Adding a subnet to existing group	$(d_1 h_1 + 1)\,k_s + k_p$	$\lceil\left(n_2 + \frac{d_2(n_2-1)}{d_2-1} + 1\right) k_s\rceil$
Evicting a member from subnet	0	$(d_2 h_2 - 1)\,k_s$
Evicting a subnet	$(d_1 h_1 - 1)\,k_s$	0

Table 1. Communication Cost in Tiered Tree Based Key Management with LKH algorithm.

RP root	SKC	RP	Member
$\lceil \frac{(d_1 n_1 - 1)}{d_1 - 1} k_s + n_1 k_p \rceil$	$\lceil \frac{(d_2 n_2 - 1)}{d_2 - 1} k_s + 2 \rceil$	$\lceil h_1 + 2 \rceil$	$\lceil h_2 + 2 \rceil$

Table 2. Storage Cost in Tiered Tree Based Key Management with LKH algorithm.

6. COST ANALYSIS

We compute the cost for communication and storage for the basic key tree scheme: LKH in the overlay and in each subnet. The results are derived by applying the cost metrics of the basic LKH to the RP tree and the SN tree, and by aggregating the two. Table 1 shows the communication overhead for the RP tree and SN tree individually, while Table 2 gives the total storage cost in the framework, using basic LKH algorithm. n is the total number of members in the group; n_1 is the number of RPs, n_2 is the number of members in each subnet; d_1, h_1 are respectively the degree and height of the RP tree; d_2, h_2 are respectively the degree and height of the SN tree; k_p is the length of a public key and k_s is the length of a symmetric key. The figures for the communication cost are only approximate; we do not rigorously consider the fact that the root of the RP tree itself is a group member. The storage costs consider that the RP root stores the public keys of all subscribed RPs, though the public keys are not needed except for the initial setup.

A comparison of the cost in the proposed framework, with respect to the protocols mentioned in section 2, can be found in [Roy-Chowdhury, 2003]. The comparison shows that in most of the cases the proposed framework fares at least as well or better than the other protocols for various metrics.

7. SIMULATION

We have verified the validity and feasibility of our framework through simulations using OPNET Modeler 9.0 [Opnet, 2002]. We used the multicast sim-

ulation setup from [Roy-Chowdhury, 2003] and added our security features to it.

We consider three multicast groups in the network, each spread across 31 subnetworks. Each group has 10 sources in 10 subnetworks, and 1075 receivers spread across all the subnetworks. For encryption, we simulate use of 64 bit symmetric keys and 1024 bit public keys. The simulation was run for 300 seconds.

MARS selected 3 different RPs as the root of the RP trees for the three groups. These RPs are leaves in the RP trees for the groups for which they are not the RP tree root. Thus in our framework, the key management in the overlay can be distributed among different RPs for different groups.

The savings in terms of bytes of key information sent per second is illustrated in figure 4, which compares the total key information sent for all the groups in the RP trees and all the SN trees, to the total key information sent on the RP trees only. As the graph shows, the resource savings on the satellite links is substantial using the tiered tree scheme. Even though the group dynamics are high, the amount of message exchanges are very few in the RP tree. This is because the RPs remain subscribed to the RP tree as long as there is at least one member in its local subnetwork sending to or receiving from the group; the frequency of joins and leaves in the subnetwork is transparent to the RP tree. This is precisely our intention, to minimize the cost of message exchanges over the satellite links. The figure also illustrates another important point of our key management scheme, namely, *1-affects-n* scalability. The effect of frequent member joins and leaves in one subnetwork remains localized within the subnetwork, and does not affect the group dynamics in other subnetworks.

8. CONCLUSION

In this paper we have proposed a framework for key management for secure multicast in a wide-area hybrid satellite network. Our design is scalable and efficient and well suited for the unique network architecture that we consider. The framework is essentially a generic design; different types of key management algorithms can be applied in each logical grouping. We considered tree based algorithms due to their scalability and robustness for large groups. However, if the subnetworks in a group are limited and remain static, then GKMP might be a good candidate for the overlay. Likewise, if total members within a subnetwork are small, then we can use GKMP or HFK in a subnet, for example. We intend to analyze the costs and tradeoffs involved in using different key generation schemes in the proposed framework.

The generation of the datahiding key for a group requires the SKCs of all subnetworks in the group to be in agreement about the datahiding key. We

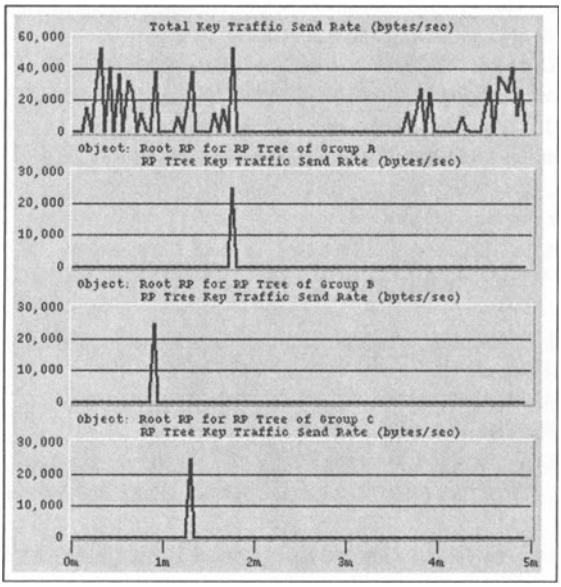

Figure 4. Tiered Tree Framework - Many-to-Many: Total Key Traffic vs. RP Tree Traffic for 3 Groups (Y-axis shows the traffic in bytes/sec; X-axis is the simulation duration in minutes).

have not considered the key management for the datahiding key, since that is a one time message exchange. A simple mechanism to do this is for the SKC in the root RP subnetwork to generate the key and send it to the SKCs in the other subscribed subnetworks; the generating SKC can know of the other subnetworks in a message from the root RP.

In other future work, we plan to investigate mechanisms for source authentication, with suitable modifications for groups operating in the hybrid broadcast network.

9. ACKNOWLEDGMENTS

The first author thanks Dr. Virgil Gligor and Vijay Bharadwaj for helpful discussions on the security framework, and Nalini Bharatula for help with the simulations. Thanks are also due to the anonymous reviewers for their valuable feedback.

The research work reported here was supported by a grant from Lockheed Martin Global Telecommunications, through Maryland Industrial Partnerships under contract number 251715, and by NASA under award number NCC 8235.

References

Armitage, G. (1997). "IP Multicasting over ATM Networks". *IEEE Journal on Selected Areas in Communications*, 15(3):445–457.

Berkovits, S. (1991). "How To Broadcast A Secret". *Advances in Cryptology - EUROCRYPT '91, Lecture Notes in Computer Science, LNCS*, 547:535–541.

Burmester, M. and Desmedt, Y. (1994). "A Secure and Efficient Conference Key Distribution System". *Advances in Cryptology - EUROCRYPT '94, Lecture Notes in Computer Science.*

Canetti, R., Garay, J., Itkis, G., Micciancio, D., Naor, M., and Pinkas, B. (1999). "Multicast Security: A Taxonomy and Some Efficient Constructions". *Proceedings of INFOCOMM '99.*

Chiou, G. and Chen, W. (1989). "Secure Broadcasting Using the Secure Lock". *IEEE Transactions on Software Engineering*, 15(8).

Deering, S.E., Estrin, D., Farinacci, D., Jacobson, V., Liu, C-G, and Wei, L. (1996). "The PIM Architecture for Wide-Area Multicast Routing". *IEEE/ACM Transactions on Networking*, 4(2):153–162.

Fiat, A. and Naor, M. (1994). "Broadcast Encryption". *Advances in Cryptology - CRYPTO '93, Lecture Notes in Computer Science, LNCS*, 773:480–491.

Harney, H. and Muckenhirn, C. (1997). *"Group Key Management Protocol (GKMP) Architecture"*. Internet RFC 2094.

Kim, K. S., Kim, S. J., and Won, D. H. (1996). "Conditional Access System Using Smart Card". *Proc. of JCCI'96, The 6th Joint Conference on Communication and Information*, pages 180–183.

Matsumoto, T. and Imai, H. (1988). "On the KEY PREDISTRIBUTION SYSTEM: A Partical Solution to the Key Distribution Problem". *Advances in Cryptology - CRYPTO '87, Lecture Notes in Computer Science, LNCS*, 293:185–193.

Mittra, S. (1997). "Iolus: A Framework for Scalable Secure Multicasting". In *Proceedings of ACM SIGCOMM'97*, pages 277–288.

Opnet (2002). Opnet Modeler 9.0. http://www.opnet.com/products/modeler/home.html.

Perrig, A., Song, D., and Tygar, J.D. (2001). "ELK, a New Protocol for Efficient Large-Group Key Distribution". *Proceedings of IEEE Security and Privacy Symposium S&P2001.*

Poovendran, R. (1999). *"Key Management for Secure Multicast Communication"*. PhD thesis, University of Maryland College Park.

Rodeh, O., Birman, K., and Dolev, D. (2000). "Optimized Group Rekey for Group Communication Systems". *Proceedings of Network and Distributed System Security Symposium (NDSS'00).*

Roy-Chowdhury, A. (2003). "IP Routing and Key Management for Secure Multicast in Satellite ATM Networks". Master's thesis, Univeristy of Maryland College Park.

Steer, D., Strawczynski, L., Diffie, W., and Wiener, M. (1990). "A Secure Audio Teleconference System". *Advances in Cryptology - CRYPTO '88, Lecture Notes in Computer Science, LNCS*, 403:520–528.

Steiner, M., Tsudik, G., and Waidner, M. (2000). "Key Agreement in Dynamic Peer Groups". *IEEE Transactions on Parallel and Distributed Systems*, 11(8):769–780.

Wallner, D., Harder, E., and Agee, R. (1999). *"Key Management for Multicast: Issues and Architectures"*. Internet RFC 2627.

Wong, C. K., Gouda, M., and Lam, S. S. (2000). "Secure Group Communications Using Key Graphs". *IEEE/ACM Transactions on Networking*, 8(1):16–30.