SECURE GROUP COMMUNICATION

Ritesh Mukherjee

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfilment of the Requirements

for the Degree of Doctor of Philosophy

Concordia University

Montreal, Quebec, Canada

September 2005

Your file Votre référence
ISBN: 0-494-09968-2
Our file Notre référence
ISBN: 0-494-09968-2

ABSTRACT

Secure Group Communication

Ritesh Mukherjee, Ph.D.

Concordia University, 2005

With the advent of digital technologies and widening Internet bandwidth in recent years there has been a marked rise in new multimedia services, including teleconferencing, pay-per-view TV, interactive simulations, software updates and real-time delivery of stock market information. Multicast data distribution has been used in controlled environments to deliver such services. However, the lack of secure, accountable multicast data distribution has prevented its use in general Internet environments. Proposals for multicast security solutions so far are complex and often require trust in intermediate components or are inefficient. A secure multicast protocol suite must provide for data confidentiality, for multicast packet source authentication and for the representation of multicast security policies. In this thesis we present a robust, simple and efficient multicast key management protocol based on proxy encryption, a multicast data source authentication mechanism based on symmetric message authentication codes and a simple multicast policy representation scheme based on XML. The solutions are analyzed and compared to previously published schemes. The results show that the proposed schemes are efficient and scalable when compared to existing schemes.

ACKNOWLEDGEMENTS

I want to thank my supervisor, Dr. J. W. Atwood, who has convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. He has given me enormous freedom to pursue my own interests and provided me with guidance and steadfast encouragement to ensure that my efforts contribute to the mainstream of networking research.

Special thanks should be given to the Computer Science and Software Engineering Department and my student colleagues who helped me in many ways. Finally, words cannot express the thanks I owe to my parents and my brother for their encouragement and support.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Traditionally networks were designed to transmit data such as files from point to point. However over the last decade there has been an explosion of new Internet technologies. Today there is an increasing demand for multipoint communications (multicast) between various parties. Multicast communications have been used for emerging Internet applications such as real-time information services, pay-per-view TV, distributed interactive simulations and multi-party games.

The demand for multiparty communications has resulted in the development of scalable protocols to support these communications. In applications such as video conferencing, distance learning, large scale content distribution, distributed computing and pay-per-view TV only authorized participants should be allowed access to group communications.

Multicast has potential for reducing network resource requirements where identical content is to be delivered to hundreds or thousands of customers. However, the present model for multicast data delivery is "anyone can send – anyone can receive". This makes it difficult to restrict access to authorized participants (i.e., paying customers). Multicast has been deployed in controlled environments but it is not viable for use in the general Internet environment. Encryption techniques can be used to render the data useless for participants without valid keys. An encryption algorithm takes a group message and performs some transformations on it using a key; the key is a randomly generated number chosen from a large range of values. This process generates a cipher text. There is no easy way to recover the original message from the

cipher text other than by knowing the key. Applying such a technique one can run secure multicast sessions. This implies the generation and distribution of a group key to all valid members of the group. The messages are protected by encryption using the chosen key, which in the context of group communication is called the group key. Only those who know the group key are able to recover the original message. A group member leaving the group should not have access to group data from the time it leaves the group; this is known as perfect forward secrecy (PFS). A group member joining the group should not have access to group data sent prior to the time when it joined the group; this is known as perfect backward secrecy (PBS). To ensure forward and backward secrecy the shared group key must be changed on each membership change and securely distributed to all existing members of the group. This is referred to as group re-keying. For large groups with frequent membership changes the cost of re-keying the group can be quite substantial. It is essential that all group members receive the new decryption key before it takes effect. However the key distribution server needs to know that all the members of the group have received the key before the sender can use the new key. Therefore the server must wait for an acknowledgment message from each member. It must drop members from the service who do not respond after several retransmissions to avoid being disrupted. It is therefore advantageous to divide the key re-distribution into subgroups that are independently managed. Scalable re-keying for large and dynamic groups is thus an important and challenging problem.

Besides data confidentiality, multicast authentication is necessary to ensure the following:

- The received packets actually originated from a legitimate sender.

- The received packets were not modified while in transit.
- Packets sent by an adversary are not mistaken as legitimate packets.

Multicast authentication is important when receiving news feeds, stock quotes or when watching pay-per-view TV. These kinds of applications require low-cost, highly efficient packet authentication. Consider a TV station, such as ESPN broadcasting sporting events live. The TV station is broadcasting to thousands of receivers where people are logged in watching the telecast. The users would want to ensure that the broadcast is from ESPN rather than a malicious third party transmitting offensive material. Another scenario could be a different brokerage houses receiving stock quotes from the New York Stock Exchange (NYSE) for their agents and displaying it to their clients through their respective brokerage house websites. Also online newspapers could receive quotes from the stock exchange and display it on their websites for the general public. The brokerage house and newspapers around the world would like to ensure that the quotes they are receiving have not been tampered with. A host of other scenarios can be envisaged where multicast authentication is imperative such as online teaching, software updates, company broadcasts, etc. These services all have a one sender-multiple receiver model. In fact multicast has the maximum bandwidth saving advantage when there are thousands of receivers and very few senders. From a multicast key management perspective, many of the key management schemes use the multicast protocol itself for re-keying and this requires that the key distribution packets be authenticated otherwise an adversary may send bogus re-key packets and disrupt the service.

Multicast data confidentiality and source authentication are two important tasks necessary to encourage the use of multicast as the technology for delivering data.

In group communications using multicast to deliver packets, different levels of security may be needed depending upon the context of the application. For example a pay-per-view session may not require the same security and authentication as is required in a corporate broadcast delivering sensitive company information. Though multicast is the technology for delivering data in each of the cases they require different security settings. As multicast groups may be distributed throughout the Internet, a system administrator should have the ability to express, monitor and enforce group security policies. There is a need for a policy representation scheme for secure group communications. It should have the ability to express policies clearly and should possess the possibility of modifying policies or creating new policies depending upon the context of the application.

In this thesis we present a multicast key management protocol, a multicast data source authentication mechanism and a multicast policy representation scheme. Chapter 2 deals with multicast and the challenges associated with multicast security. Chapter 3 presents the group communication security requirements. Chapter 4 describes the existing solutions to group security, their advantages and drawbacks. Chapter 5 presents our solution to group key management, its functioning and an in-depth analysis. Chapter 6 outlines the data source authentication requirements. Chapter 7 discusses various existing approaches to data authentication in multicast. Chapter 8 presents our solution to multicast data authentication. Chapter 9 describes the multicast policy management requirements. Chapter 10 presents the conclusions and future work.

2 MULTICAST SECURITY

Multicasting provides an efficient way of delivering high data rate content over the Internet:

- Multicasting can dramatically reduce the network bandwidth that multimedia applications require.
- Servers do not require hardware upgrades in order to take advantage of multicasting.
- Clients do not require hardware upgrades in order to take advantage of multicasting.
- Because routers of recent vintage already support multicasting, enabling multicasting on a network is practical and cost-effective.

2.1 IP MULTICAST BASICS

The bulk of the traffic on today's networks is unicast. In unicast a separate copy of the data are sent from the source to each client that requests it. Unicast wastes bandwidth by sending multiple copies of the data as shown in figure 1.

Networks also support broadcasting. When data are broadcast, single copies of the data are sent to all clients on the network. Broadcast wastes bandwidth by sending the data to the whole network whether or not the data are wanted as shown in figure 2. Broadcasting can also needlessly slow the performance of client machines. Each client must process the broadcast data whether or not the broadcast is of interest.

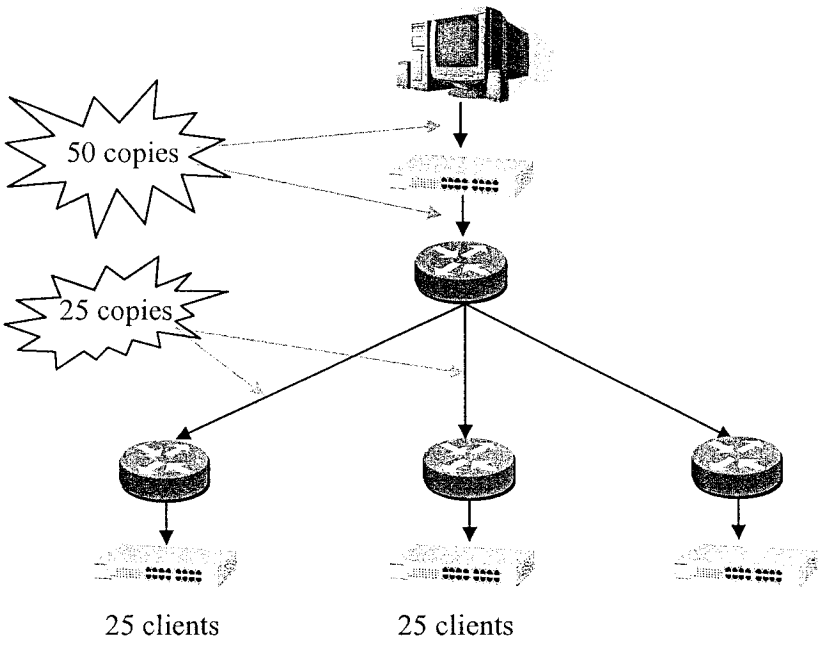

Figure 1: Unicast Transmission

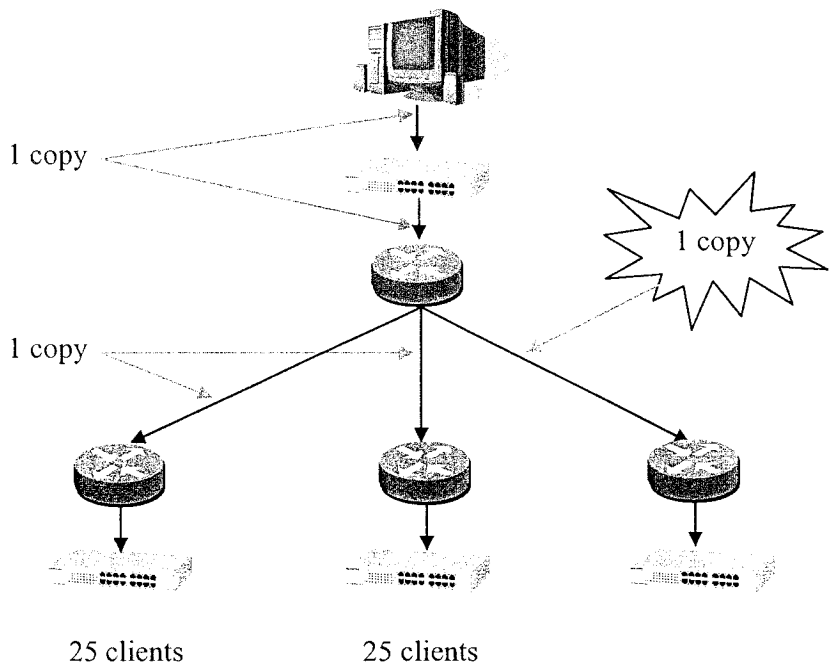

Figure 2: Broadcast Transmission

When the same data needs to be sent to only some of the nodes on the network both unicast and broadcast waste network bandwidth.

Multicast sends a single copy of the message to the clients that request it. It combines the advantages of unicast and broadcast while eliminating the weaknesses. The Internet Multicast Backbone, the *MBone*, is a virtual network consisting of those portions of the Internet that are multicast capable. These areas are connected to each other through unicast tunnels. The MBone has been used to multicast live audio and video showing Internet Engineering Task Force conferences, NASA astronauts working in space and the Rolling Stones in concert. This has demonstrated the practicality and use of multicast in sending multimedia data across the Internet.

Multicast traffic is directed to a group of hosts. These hosts comprising of senders and receivers form the multicast group. The hosts can choose whether they wish to participate in the multicast group by expressing their intent through the Internet Group Management Protocol (IGMP for IPv4 hosts) [1] or Multicast Listener Discovery (MLD for IPv6 hosts) [2]. *Multicast addresses* specify an arbitrary group of IP hosts that have joined the group and wish to receive traffic sent to this group. The *Internet Assigned Numbers Authority* (IANA) controls the assignment of IP addresses. It has assigned the Class D address space to be used for multicast. Thus all IP multicast group addresses must fall in the range 224.0.0.0 to 239.255.255.255. This refers to the group address or destination address. The source address for multicast datagrams is always the unicast source address. IGMP is used to dynamically register individual hosts in a multicast group on a particular local area network. Hosts identify group memberships by sending IGMP messages to their local multicast router. Under IGMP,

routers listen to IGMP messages and periodically send out queries to discover which groups are active or inactive on a particular subnet.

The multicast routing protocols are responsible for construction of multicast data distribution trees through the network and perform multicast forwarding as shown in figure 3. Distribution trees define the path that multicast traffic will take through the network to group members. These paths are based on source trees or a shared tree.

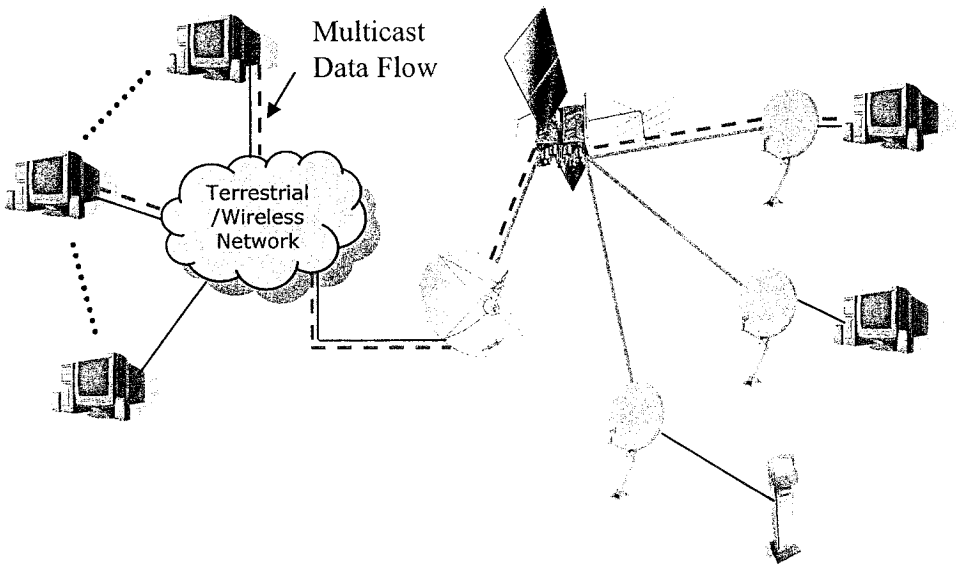

Figure 3: Multicast Transmission

The simplest method is a source tree, with its root at the source and branches forming a spanning tree throughout the network. It is also known as the shortest path tree (SPT). Source trees guarantee minimal latency, but require more network resources because a separate tree is built for every sender in the multicast group.

Shared trees use a single common root placed at some chosen point in the network. This shared root is often called a rendezvous point. Multicast sources send their traffic to the rendezvous point, which forwards traffic down the shared tree to all group members. Shared trees make more efficient use of network resources, but don't necessarily use an optimal path, which may increase packet latency.

Multicast routing protocols also come in two flavours: dense mode and sparse mode. Routing protocols such as Distance Vector Multicast Routing Protocol (DVMRP) [3] and Multicast Open Short Path First (MOSPF) [4] were designed to handle dense multicast groups. Protocol Independent Multicast – Sparse Mode (PIM-SM) [5] is a sparse mode protocol that provides efficient communication between sparsely distributed group members. PIM-SM uses an explicit "join" model that blocks multicast traffic forwarding unless it is requested; routers must explicitly join a group to receive multicast traffic for that group. This prevents unnecessary flooding of multicast traffic throughout the network, resulting in greater bandwidth efficiency and multicast scalability.

PIM-SM, a shared tree protocol, also allows switchover from the rendezvous point-based tree (RPT) to the shortest path tree (SPT) model if a performance threshold is violated, offering the best of both worlds. A router will change from the RPT to the SPT if the multicast traffic it receives exceeds a predefined latency threshold, providing a quality-of-service mechanism for multicast.

IP multicast provides an efficient and simple way for enterprise network managers to distribute information and a significant incentive for service providers to deliver affordable public multicast services.

2.2 NEED FOR MULTICAST SECURITY

There are immense opportunities for a market using multicast content delivery. End users can use computers, digital video receivers, mobile phones, etc., to receive multicast content such as news broadcasts, stock quotes, movie channels, etc. The use of multicast has made it feasible to deliver high data rate content to a large number of receivers.

In spite of the advantages presented by IP multicast it is not widely deployed. One of the major obstacles in commercial IP multicast deployment is the current IP multicast model. Lack of access control and user usage information makes it difficult for a service provider to generate revenue. Internet Service Providers need to accurately identify the users and collect their usage information to generate correct billing information. For example a service provider providing IP multicast TV would like to know what and how much the users are watching so that they can be charged appropriately. Content providers need to learn about content usage information. This information could be used by content providers to set different charges for their content depending upon the popularity of the content. For example watching a popular TV serial would cost more than watching a TV serial that is less popular. They could also use this information to generate appropriate advertisement revenue.

For example placing an advertisement during a popular TV serial would cost more than placing an advertisement during a TV serial that is less popular.

To introduce access control it is necessary to establish a key management scheme, which can then be used to distribute keys to receivers. The keys are used to encrypt the multicast data being sent and only receivers with legitimate keys can view the multicast data. This would require receivers to authenticate themselves to a key server and obtain keys, which in turn will enable the ISPs to track usage information. The users would also like to ensure that the multicast data they are receiving is being sent by the actual sender and not by an adversary posing as a sender. This requires a multicast data source authentication scheme. For different devices running multicast applications to be able to interpret keys and security information, there is a need for a generic multicast policy representation scheme.

2.3 CHALLENGES OF MULTICAST SECURITY

The issue that still discourages wide deployment of IP multicast services is the lack of a suitable security mechanism to protect and authenticate the multicast data. Also there is no generic scheme for representing multicast security policies.

2.3.1 DATA CONFIDENTIALITY

For multicast data confidentiality a security association between the senders and receivers of the multicast channel is required. A security association is a set of cryptographic keys and algorithms associated with the multicast channel. There are

security associations available for unicast communications. These cannot be used in a multicast environment because of the dynamic nature of multicast groups. In unicast communications the sender and receiver are static. If any of them decide to leave the channel then the communication comes to a halt. In a multicast channel the senders and receivers are dynamic. They may join and leave the multicast group while communications must go on. The keying material associated with a multicast group must also dynamically change to ensure that users who have left the multicast group do not have access to multicast data being sent and also to ensure that users who have joined the multicast group do not have access to multicast data that was sent earlier. These conditions can vary for different multicast groups.

Multicast groups and the keying mechanism associated with the group must be independent of the size of the group as described in [6]. The main scalability pitfalls in multicast security are:

- The "one affects all" failure occurs when the action of one group member affects everybody in the group. It is not feasible to change the keys for the entire group when one member joins or leaves the group. Changing the keys for hundreds of users in a multicast group over a network would stall the network completely resulting in disruption of multicast data flow, packet losses and delays.
- The "one does not equal n" failure occurs when the group cannot be treated as a whole but must be treated as a set of individuals. It is not feasible to individually distribute the key to each member when a change occurs.

To illustrate these ideas, Figure 4 shows an example multicast tree, with a two-level hierarchy.

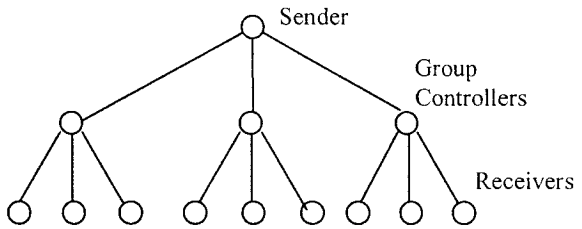

Figure 4: Example Tree

Consider the scenario shown in Figure 5, where a new member joins the group having a shared group key K_g. The simplest method is to generate a new shared group key K_g' and multicast it to the existing group members using K_g as well as unicast it to the new member using a secure channel. This causes a scalability problem because the entire group must update their shared group keys when a new member joins.

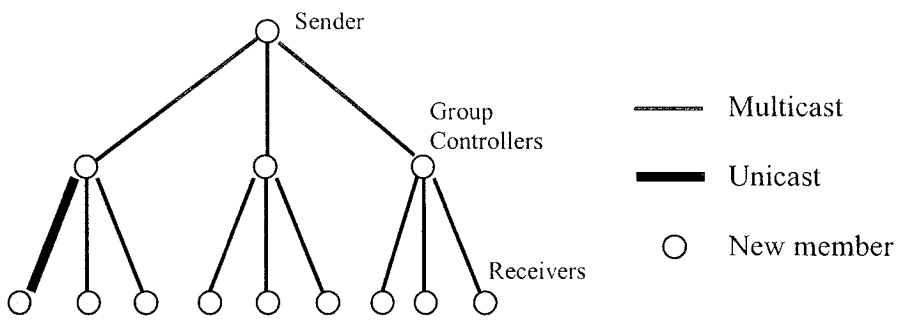

Figure 5: Member Join

Consider the scenario shown in Figure 6, where a member leaves the group. As before the shared key must be updated from K_g to K_g'. But there is no efficient method to securely distribute the new shared key to the members of the multicast group excluding the member that has just left the group because the member leaving the

group knows K_g. Therefore K_g cannot be used to encrypt K_g'. The simplest method is to send K_g' to all the group members by separately unicasting it to each member over secure channels. This method corresponds to the single centralized controller, and is totally unacceptable for large and dynamic groups.

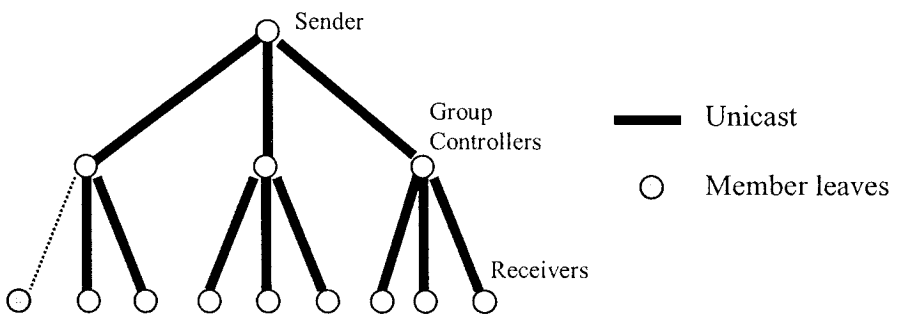

Figure 6: Member Leave

Thus the challenges associated with any multicast key management scheme are as follows:

- Generate new keys when required
- Securely distribute the keys to the required group members
- Re-keying should be possible with minimum number of messages
- The solution must be simple, scalable and reliable.
- It should not place complete trust in the intermediate nodes

2.3.2 DATA SOURCE AUTHENTICATION

Multicast authentication is a challenging problem because of the large number of participants involved in the communication and the need to authenticate a large number of packets any of which may be lost. The simplest solution to authentication

is when each packet is signed by the sender. The receiver could verify the signature and discard packets whose signatures were not verified. However this solution is unacceptable because of the repeated use by the sender of a computationally expensive sign primitive for each packet and the communication overhead caused by the addition of a signature to each packet. Another solution to authentication is to use hash based message authentication codes (HMACs) as in unicast transmission. This does not work well in a multicast setting if small overhead is required and time synchronization between sender and receivers is difficult to maintain. The use of simple symmetric message authentication codes (MACs) is unacceptable as it would enable any of the receivers to impersonate the sender. Because the stream of multicast packets is implemented using UDP and the loss of packets is acceptable in many applications such as pay-per-view TV, the authentication scheme must tolerate packet loss. Also participants may join at any point in time and they should be able to begin authenticating packets starting from any packet. It is possible to compute signatures in advance and to buffer them for content that is already available, such as applications showing movies over the Internet. However real time multicast applications such as stock and news feeds cannot use signatures that take a long time to compute.

An efficient authentication scheme must

- be able to ensure that the received packets actually originated from a legitimate sender
- be able to ensure that the received packets were not modified while in transit
- be able to ensure that packets sent by an adversary are not mistaken as legitimate packets
- work with both real time and previously available content

- be able to provide authentication from any packet onwards despite losses in the network
- reduce the communication overhead by adding a small number of bytes per packet
- not be computationally very expensive

2.3.3 MULTICAST POLICY REPRESENTATION

The availability of new devices with broader networking capabilities, powerful processors and large memories has influenced the way one thinks about security and authentication. Dynamic applications with different requirements and changing security mechanisms need not necessarily require overhauling the entire framework or reworking the business logic or functional requirements. Security policies define the access controls and permissions of different resources making it possible to update and modify the rules depending upon the context of the application. Communication between different parties in the network takes place when all parties can meet the security policies prescribed for the communication. In certain cases network devices may be forced to modify their security policies or to abort communication.

A group security policy defines the security relevant behaviours, access control parameters and security mechanisms used to implement the group [7]. A set of keys, members, protocols and algorithms form a security context. A security policy defines how these policies must be derived and a policy framework defines the entities and protocols to define, interpret, negotiate, and distribute a secure group policy [8]. Security in group communications is established by using encryption, which renders

the data useless for network nodes without proper decryption keys. Multicast groups may have different threat models. For example some multicast groups may require that users who join the group after a certain time are not able to access multicast data already sent and users who have left the multicast group are not able to access data sent afterwards. This requires a key change at every membership change. Some other types of multicast groups may not have such stringent rules and may not require key changes at every membership change. The multicast policy representation language must be flexible to accommodate a large variety of such policies.

Any policy representation scheme must:

- have the ability to express policies clearly
- possess the possibility of modifying policies or creating new policies depending upon the context of the application
- be flexible and adaptable
- not require development of tools or modules to create, edit or parse policies
- be easily understood by administrators and users

2.4 MSEC ARCHITECTURE

The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. The actual technical work of the IETF is done in its working groups, which are organized by topic into several areas (e.g., routing, transport, security, etc.). The purpose of the Multicast Security Working Group (MSEC WG) is to standardize protocols for

securing group communication over internets, and in particular over the global Internet.

MSEC has defined an architecture [9] concerned with the securing of large multicast groups. This architecture is "end to end", and does not require multicast routing protocols to participate in this architecture. Inappropriate routing may cause denial of service to application layer groups conforming to this architecture. However the routing cannot affect the authenticity or secrecy of group data or management packets. The multicast routing protocols could themselves use this architecture to protect their own multicast and group packets. The architectural design is defined in the context of a Reference Framework. This Reference Framework is used to classify functional areas, functional elements, and interfaces. The aim of the Reference Framework is to provide some general context around the functional areas, and the relationships between the functional areas. Note that some issues span more than one functional area. In fact, the framework encourages the precise identification and formulation of issues that involve more than one functional area or those that are difficult to express in terms of a single functional area.

The need for solutions to be scalable to large groups across wide geographic regions of the Internet requires the elements of the framework to also function as a distributed system. Figure 7 shows how distributed designs supporting large group scalability fit into the Reference Framework.

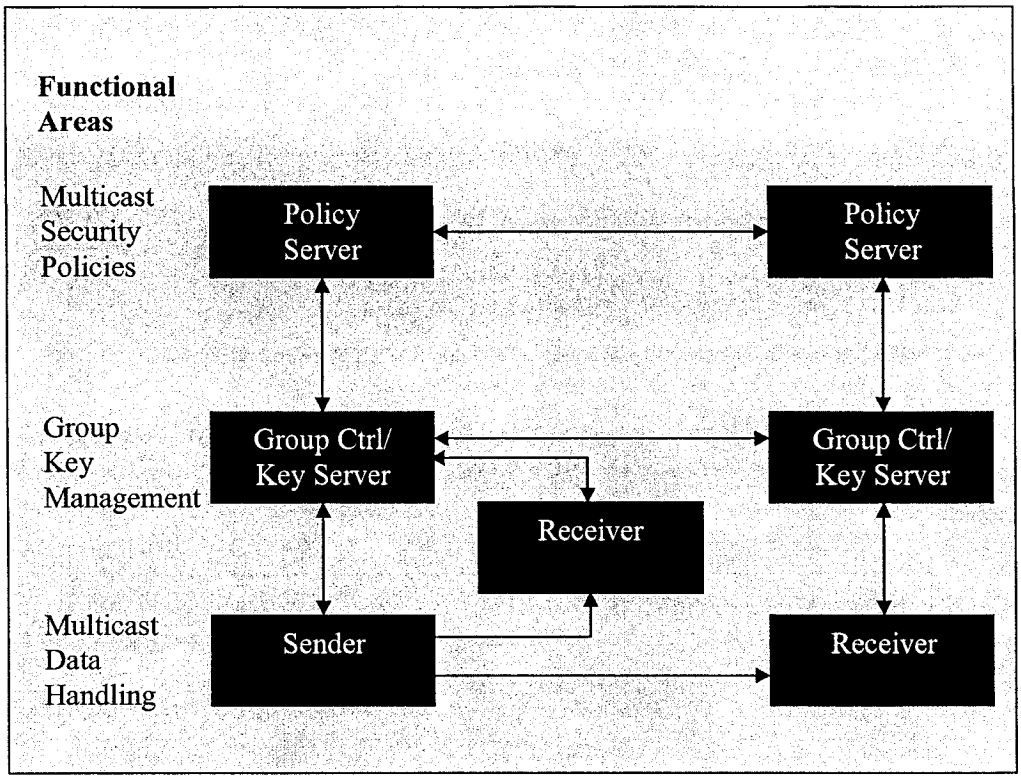

Figure 7: Distributed Multicast Security Reference Framework

The Group Controller and Key Server (GCKS) represent both the entity and functions relating to the issuance and management of cryptographic keys used by a multicast group. The Key Server (KS) and the Group Controller (GC) have somewhat different functionality and may in principle be regarded as separate entities. The GCKS also conducts user-authentication and authorization checks on the candidate members of the multicast group. In a distributed design the GCKS entity interacts with other GCKS entities to achieve scalability in the key management related services. GCKS entities will require a means of authenticating their peer GCKS entities, a means of authorization, and a means of interacting securely to pass keys and policy. The Policy Server represents both the entity and functions used to create and manage security policies specific to a multicast group. The Policy Server interacts with the GCKS

entity in order to install and manage the security policies related to the membership of a given multicast group and those related to keying material for a multicast group. Policy Servers must interact with each other securely to allow the communication and enforcement of policies across the Internet. Two Receiver boxes are displayed corresponding to the situation where both the Sender and Receiver employ the same GCKS entity (centralized architecture) and where the Sender and Receiver employ different GCKS entities (distributed architecture). In the distributed design, all Receivers must obtain identical keys and policy. Each member of a multicast group may interact with a primary GCKS entity (e.g., the "nearest" GCKS entity, measured in terms of a well-defined and consistent metric). Similarly, a GCKS entity may interact with one or more Policy Servers, also arranged in a distributed architecture.

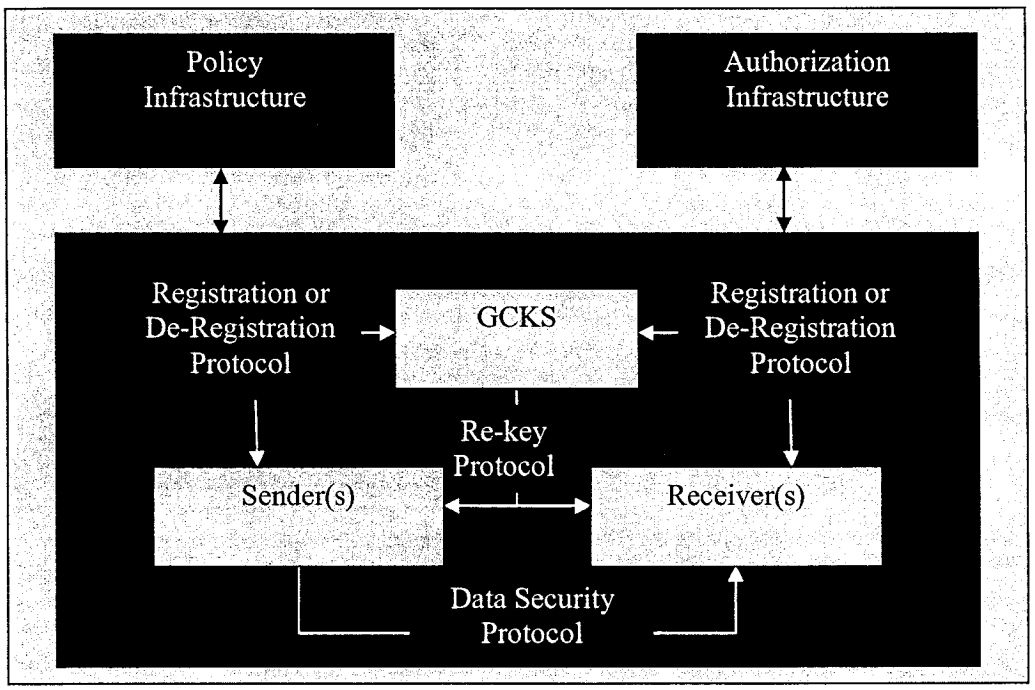

Figure 8: Group Security Association Model

MSEC also defines the common architecture of Multicast Security Key Management Protocols [10] to support variety of application, transport, and network layer security protocols. The framework and guidelines permit a modular and flexible design of Group Key Management (GKM) protocols for a variety of different settings that are specialized to applications needs.

Figure 8 depicts the overall design of a GKM protocol. Each group member, sender or receiver, uses the registration protocol to get authorized and authenticated access to a particular Group, its policies, and its keys.

The design achieves scalable operation by

- allowing the de-coupling of authenticated key exchange in a registration protocol from a re-key protocol
- allowing the re-key protocol to use unicast push or multicast distribution of group and data keys as an option,
- allowing all keys to be obtained by the unicast registration protocol, and
- delegating the functionality of the GCKS among multiple entities, i.e., to permit distributed operation of the GCKS.

Any solution to group key management protocol for data confidentiality, authentication and security policy management must adhere to these architectures to ensure smooth operation with the multicast protocol being used to deliver data and to take advantage of member authentication schemes.

3 GROUP COMMUNICATION SECURITY REQUIREMENTS

A group key management protocol helps to ensure that only members of a secure group can gain access to group data (by gaining access to group keys) and can authenticate group data [9]. The goal of a group key management protocol is to provide legitimate group members with the up-to-date cryptographic state they need for secrecy and authentication. Multicast applications, such as video broadcast and multicast file transfer, typically have the following key management requirements. Note that the list is neither applicable to all applications nor exhaustive.

- Group members receive security associations that include encryption keys, authentication/integrity keys, cryptographic policy that describes the keys, and attributes such as an index for referencing the security association (SA) or particular objects contained in the SA.
- In addition to the policy associated with group keys, the group owner or the Group Controller and Key Server (GCKS) may define and enforce group membership, key management, data security, and other policies that may or may not be communicated to the entire membership.
- Keys will have a pre-determined lifetime and may be periodically refreshed.
- Key material should be delivered securely to members of the group so that they are secret, integrity-protected and verifiably obtained from an authorized source.
- The key management protocol should be secure against replay attacks and Denial of Service (DoS) attacks.

- The protocol should facilitate addition and removal of group members. Members who are added may optionally be denied access to the key material used before they joined the group, and removed members may optionally lose access to the key material following their departure.
- The protocol should support a scalable group re-key operation without unicast exchanges between members and a Group Controller and Key Server (GCKS), to avoid overwhelming a GCKS managing a large group.
- The protocol should be compatible with the infrastructure and performance needs of the data security application, such as the IPsec security protocols AH and ESP, and/or application layer security protocols such as SRTP [11].
- The key management protocol should offer a framework for replacing or renewing transforms, authorization infrastructure, and authentication systems.
- The key management protocol should be secure against collusion among excluded members and non-members. Specifically, collusion must not result in attackers gaining any additional group secrets than they are authorized to receive. In other words, combining the knowledge of the colluding entities must not result in revealing additional group secrets.
- The key management protocol should provide a mechanism to securely recover from a compromise of some or all of the key material.
- The key management protocol must be efficient. It must have low bandwidth consumption, low computational workload and small code size.

4 TAXONOMY OF KEY MANAGEMENT APPROACHES

The different approaches to group re-keying can be broadly classified into four main classes: centralized methods, distributed subgroup methods, distributed methods and hierarchical methods.

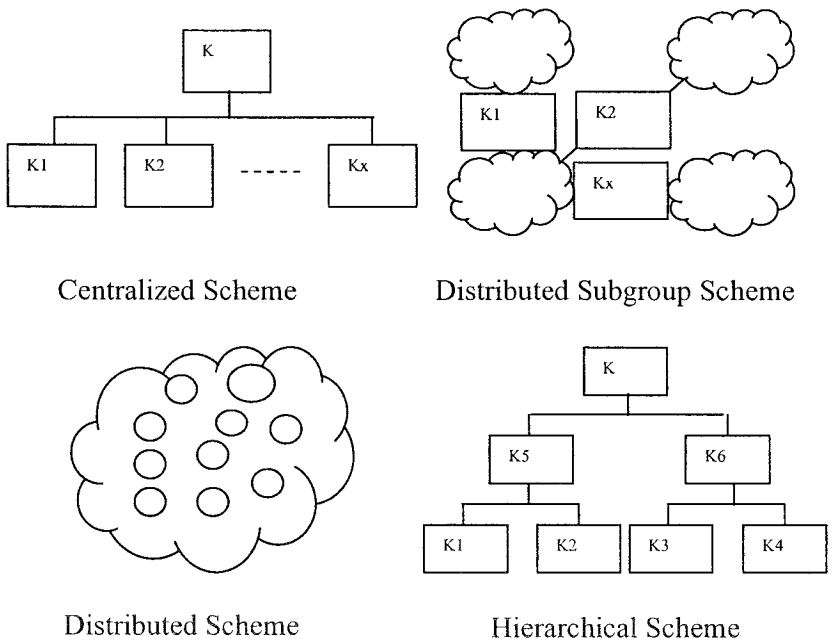

Figure 9: Key Management Approaches

In the centralized method a single entity is controlling the entire group. In this scheme the central entity is a critical point of failure. The failure of this central entity causes the entire group to become insecure because keys are no longer being generated or distributed. There is also the issue of scalability as the group may be too large for a single entity to handle. In the distributed subgroup method the large group is split into small subgroups. Different controllers are used to manage each subgroup, minimizing the problem of concentrating the work on a single place. In the distributed method there is no group controller. All members can perform access control. The group key

is generated in a contributory fashion where all members contribute their own share to the computation of a group key. Although this achieves a high scalability and fault-tolerance, it may not be safe to leave to any member to generate new keys since key generation requires secure mechanisms that may not be available to all members. Hierarchical methods organize a structured hierarchy of keys along the distribution tree thereby reducing the time and overhead involved in a re-key operation. They require the nodes to maintain a large number of keys and distributing specific keys to specific nodes makes the protocol complex. The following section discusses some of the popular key management approaches:

The Group Key Management Protocol (GKMP) [12] [13] enables the creation and maintenance of a group key. In this approach, the Key Distribution Centre (KDC) helped by the first member to join the group creates a Group Key Packet (GKP) that contains a group traffic encryption key (GTEK) and a group key encryption key (GKEK). When a new member wants to join the group, the KDC sends it a copy of the GKP. When a re-key is needed, the GC generates a new GKP and encrypts it with the current GKEK. As all members know the GKEK, there is no solution for keeping forward secrecy when a member leaves the group except to recreate an entirely new group without that member.

Scalable Multicast Key Distribution (SMKD) [14] proposes a scheme to use the trees built by the Core Based Tree (CBT) multicast routing protocol to deliver keys to a multicast group. Any router in the path of a joining member from its location to the primary core can authenticate the member since the router is authenticated with the primary core. This scheme assumes that CBT (a protocol that has not seen wide

acceptance) is deployed. Furthermore, there is no solution for forward secrecy other than to recreate an entirely new group without the leaving members.

Iolus [6] is a framework with a hierarchy of agents that splits the large group into small subgroups. A Group Security Agent (GSA) manages each subgroup. The GSAs are also grouped in a top-level group that is managed by a Group Security Controller. Iolus uses independent keys for each subgroup and the absence of a general group key means membership changes in a subgroup are treated locally. It means that changes that affect a subgroup are not reflected in other subgroups. In addition, the absence of a central controller contributes to the fault-tolerance of the system. If a subgroup controller (namely GSA) fails, only its subgroup is affected. Although Iolus is scalable, it has the drawback of affecting the data path. There is a need for translating the data that go from one subgroup to another. The data have to be decrypted and encrypted with a different key. The GSA has to manage the subgroup and perform the translations needed. The GSA may thus become a bottleneck. The use of encryption indirection can enable this latency to be constant and independent of the packet lengths. Instead of having the sender directly encrypt the data with the group key for that area, the sender generates a random key on a per-transmission basis and uses this key to encrypt the data. It then includes this key with the data, encrypted with the group key for that area. In this way decrypting and re-encrypting the packet is reduced to decrypting and re-encrypting the random key. However this makes it susceptible to transient security breaches. This requires the senders to unicast the data to the GSA with a unique key they share with the GSA. The GSA then decrypts the data, re-encrypts it with the group key and then multicasts it to the group. This adds an extra

cost of having the unicast the data, requires one decryption and re-encryption in the data path and introduces an overhead caused by adding a random key to every packet.

In the Logical Key Hierarchy (LKH) [15] approach, a KDC maintains a tree of keys. The nodes of the tree hold key encryption keys. The leaves of the tree correspond to group members and each leaf holds a KEK associated with that one member. Each member receives and maintains a copy of the KEK associated with its leaf and the KEKs corresponding to each node in the path from its parent leaf to the root. The key held by the root of the tree is the group key. The scheme is complex and requires nodes to maintain large number of keys.

Mykil [16], which is a combination of subgroup based and key hierarchy based schemes, requires each packet to be decrypted and re-encrypted by the Area Controllers. The advantage of Mykil is that it provides support for mobile group members and fault tolerance. One Way Functions trees [17] are complex and require the nodes to maintain a large number of keys. SAKM [18] uses adaptive clustering of subgroups. The scheme is complex. It requires the protocol to work as a centralized scheme when real time data are being transmitted resulting in scalability problems. Kronos [19] is driven by periodic re-keying after a time period instead of a membership change. It requires different area controllers to have synchronized clocks and to agree on a re-key period.

Although a number of mechanisms have been proposed for protecting data in group communications very few of them are targeted at a large group setting with highly dynamic membership without third party trust, and if they do, they are complex and

inefficient. Most existing protocols for secure multicasting are limited to distributing session keys in static and/or small groups. A few, which deal with group key distribution in a large group with frequent membership changes, suffer from high computational complexity, high level of trust in network components and the necessity for group members to interoperate for the generation of a group-wide secret.

Most multicast applications are highly dynamic and have a single sender with a very large number of receivers. An efficient multicast group key management solution must be able to work efficiently in this scenario. In addition to distributing keys efficiently, the performance must be comparable to or better than existing schemes. It must not place a high level of trust in network components and must provide access for perfect forward and backward secrecy.

An ideal key management scheme must combine advantages of different schemes and remove the disadvantages. It must divide the distribution tree into subgroups making it highly scalable. However the mechanism must not depend upon any subgroup controller and should be able to recover from the failure of any group controller. The subgroup controllers should not have to decrypt and re-encrypt packets. It must provide compromise recovery and it must be easy to add or remove new features to it. It must use dynamic aggregation to reduce the number of messages thus minimizing bandwidth overhead. In the following chapter we present SIM-KM (Scalable Infrastructure for Multicast Key Management) which efficiently satisfies the requirements of a group key management protocol.

5 SCALABLE INFRASTRUCTURE FOR MULTICAST KEY MANAGEMENT (SIM-KM)

Scalable Infrastructure for Multicast Key Management (SIM-KM) [20] [21] uses a combination of the key hierarchy and subgroup based schemes. It retains their advantages and removes their disadvantages. This new approach, using proxy encryption [22], is scalable, does not require revealing the decryption keys to the group controllers and is simple to implement and maintain. SIM-KM also employs other techniques such as group controllers with varying trust levels and different responsibilities. The scheme also employs dynamic batching techniques to reduce the number of control messages.

5.1 OVERVIEW

Figure 10 illustrates the basic framework for transmission and key control, focusing on a single sender and receiver. The figure shows the flow of multicast data from the senders to the receivers. The participants obtain the keys from the group controller, which performs the required key management functions. When the actual distribution tree is hierarchical, there may be several group controllers, each associated with a subtree of the distribution graph. The key control and (distributed) group controller(s) can be considered as forming a logical control plane for control actions.

The framework addresses the problem of having to change the keys for the entire group by doing away with the notion of having a single secure distribution tree for the entire multicast period.

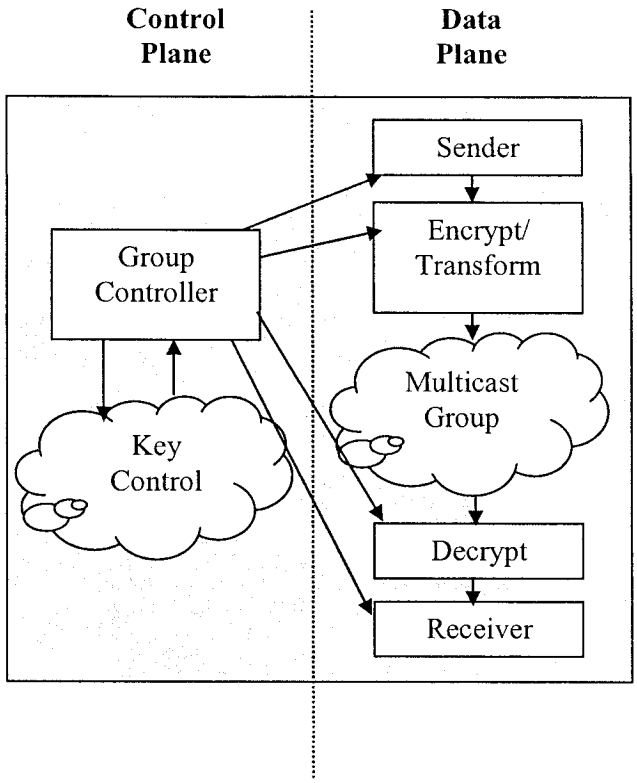

Figure 10: Basic Framework

The secure distribution tree can be divided into "subgroups" as and when required. It however maintains the distribution tree structure and periodically refreshes the key to form a single distribution tree to avoid the problem of changing the cipher text continuously in the "lower" parts of the tree. For this purpose trusted nodes in different parts of the network are designated as group controllers. These group controllers have the authority to transform the cipher text using the proxy encryption scheme. Thus they can change the cipher text but do not have the ability to see the clear text message or the ability to derive the keys from the message or the transformation algorithm. If a group controller finds out that another group controller sending it multicast data has started performing a transformation it stops performing another transformation itself and forwards JOIN and LEAVE messages received from

participants to that group controller. Senders and receivers continue to send and receive as they normally would. They join the group controller nearest to them in the network. Thus there is a hierarchy of subgroups. The root of the distribution tree encrypts the data using the algorithm and the receivers decrypt the data using the algorithm. When a transformation is applied by an intermediate node the receiving nodes are given the proxy encryption key and the intermediate node applies the transformation to modify the cipher text using the algorithm. The receivers can now decrypt the data using the decryption algorithm with the proxy decryption key. The success of the key management scheme thus depends on the ability of the group controllers to distribute the proxy decryption key to the required participants and the proxy encryption key to the required intermediate nodes. Figure 11 illustrates the functioning of SIM-KM.

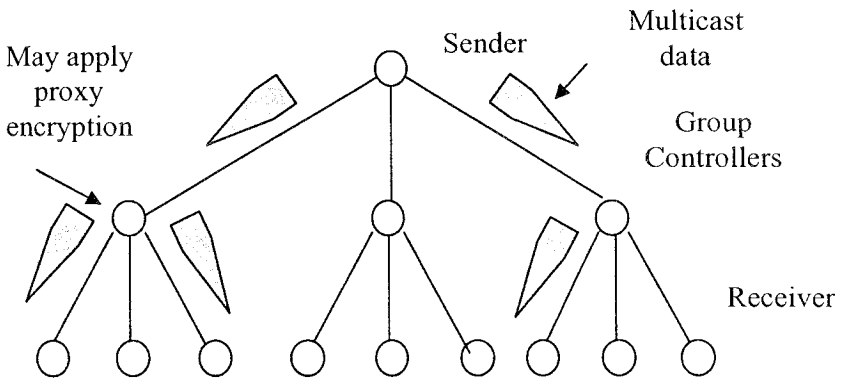

Figure 11: Functioning of SIM-KM

Proxy functions extend public key encryption algorithms such as RSA. However the method with which the asymmetric proxy encryption technique has been used in SIM-KM differs from other asymmetric key techniques. In other schemes using asymmetric keys the encryption key is freely available and the decryption key is kept a secret so any node can send a secret message to the node with the decryption key. In

SIM-KM the encryption key is kept a secret and the decryption key is made available to a selected group of receivers. It may also be possible to take advantage of encryption indirection as is done by Iolus [6]. In this method a random key can be generated and used to encrypt the data. The group key is used to encrypt the random key. The random key is sent along with the data. This requires the intermediate nodes to only transform the random key instead of having to transform the data. Iolus must perform a decryption and re-encryption even if it uses indirection whereas our scheme does not require such an operation.

5.2 PROXY ENCRYPTIONS

Proxy Functions can convert cipher text for one key into cipher text for another without revealing secret decryption keys or clear text messages. This allows a non-trusted third party to convert between cipher texts without access to the clear text message or to the secret component of the old key or new key. In simple terms, the key generation algorithm generates keys for encrypting the multicast data. The senders encrypt the multicast data using the traffic encryption algorithm and the receivers decrypt the received cipher text using the traffic decryption algorithm. Whenever an intermediary entity wants to change the cipher text it uses the traffic to proxy changing algorithm. The receiver then has to use the proxy decryption algorithm to recover the original clear text.

SIM-KM uses asymmetric proxy functions and keys. The sender has the encryption key, the intermediate nodes have the proxy transformation key and the receivers have the decryption key. El Gamal, RSA or identity based encryption schemes may be used

as proxy encryption schemes [23]. It should be noted that SIM-KM does not use the keys as is done in a public key encryption scheme where the encryption key is made freely available to anyone who wants to send data to a particular receiver. In SIM-KM the encryption key is kept a secret and is known only to senders of the multicast data. To ensure that it is not mistaken for a public key encryption scheme we refer to it as an asymmetric scheme. Here an encryption scheme based on El Gamal encryption is discussed. For certain traffic encryption approaches (including El Gamal) collusion between the intermediate entity and receivers can permit discovery of the encryption key. This can be avoided by giving a part of the decryption key to the sender and requiring it to perform a partial decryption before sending the data. This ensures that there is no possibility of deriving the encryption key as shown in the following example:

Let p be a prime, and g be a generator of $Z_p = \{1, \ldots, p - 1\}$. The private key x is an integer between 1 and p – 2. Let $y = g^x \bmod p$. The public key for El Gamal encryption is the triplet (p, g, y). To encrypt a plaintext M, a random integer k relatively prime to p – 1 is selected, and the following pair is computed:

$$a \leftarrow g^k \bmod p$$

$$b \leftarrow My^k \bmod p$$

The cipher text C consists of the pair (a, b) computed above. The decryption of the cipher text C = (a, b) in the El Gamal scheme, to retrieve the plain text M is simple:

$$M \leftarrow b/a^x \bmod p$$

In the above expression, the "division" by a^x should be interpreted in the context of modular arithmetic, that is, M is multiplied by the inverse of a^x in Z_p. The correctness of the El Gamal encryption scheme is easy to verify.

$$b/a^x \bmod p = My^k(a^x)^{-1} \bmod p$$

$$= M\, g^{xk}(g^{kx})^{-1} \bmod p$$

$$= M$$

In the Proxy El Gamal encryption scheme the private key is split into x_1 and x_2 such that $x = x_1 + x_2$. This split can be made when required and there can be a very large number of such possible splits resulting in different values of x_1 and x_2. The sender is given x_1. The sender transforms the data using x_1 after encryption. This is done to ensure that no collusion of intermediate nodes or receivers will succeed in obtaining the decryption key x. Hence there is no possibility of using the decryption key and other information to obtain the encryption key. This ensures that the receivers are unable to impersonate the sender. x_2 is further split into x_3 and x_4 such that $x_2 = x_3 + x_4$ when a membership change occurs and the decryption key for a part of the multicast data distribution tree has to be changed. The traffic to proxy changing algorithm receives x_3 and the receiver receives x_4. The traffic to proxy changing function and the decryption function are similar to the original decryption function under x_3 and x_4 respectively. The sender performs a partial decryption given by

$$M1 \leftarrow b/a^{x1} \bmod p$$

The traffic to proxy changing function thus performs a partial decryption given by

$$M2 \leftarrow M1/a^{x3} \bmod p$$

The decryption function performs the decryption by performing

$$M \leftarrow M2/a^{x4} \bmod p$$

The correctness of the Proxy El Gamal encryption function is easy to verify.

$$M2/a^{x4} \bmod p = (M1/a^{x3} \bmod p)/a^{x4} \bmod p$$

$$= (b/a^{x1} \bmod p)/\, a^{x3\,+\,x4} \bmod p$$

$$= (b/a^{x1} \bmod p)/\, a^{x2} \bmod p$$

$$= b/(a^{x1 + x2}) \bmod p$$

$$= b/a^{x} \bmod p$$

These proxy encryption schemes have been shown to be as secure as the original schemes [23].

5.3 PROTOCOL DETAILS

In SIM-KM group controllers may be either trusted or non-trusted. Group controllers that are trusted have the authority to perform most operations. They have access to the group lists and have the ability to authenticate and give access to participants. They may also perform operations on data. They are sparsely spread in the network. As they can perform control operations they are known as Control Group Controllers (CGCs). A detailed description of their functioning is provided in section 5.4.

Group controllers that are non-trusted have limited functionality. Their main task is to transform data. They are placed near the edges of the multicast tree so that the effects of changes in group composition may be localized to a small area. They may only perform operations on data and are known as Data Group Controllers (DGCs). A detailed description of their functioning is provided in section 5.4.

Nodes in the network may be designated CGCs or DGCs by the administrator. They can also be dynamically configured during operation. Any node in the network can be configured as a DGC as they do not get access to keys or clear text data or access lists. This dynamic design makes SIM-KM highly scalable. The concept of key based schemes is used to avoid having to change keys for the entire group. Group

controllers have the authority to transform the cipher text using the proxy encryption scheme. They can change the cipher text but they do not have the ability to see the clear text messages or the ability to derive the keys from the message or the transformation algorithm. If a group controller notices that another group controller upstream in the distribution tree is performing a transformation, it stops doing the transformation itself and starts forwarding JOIN and LEAVE messages received from participants to that group controller. The key is periodically refreshed to avoid having to change the cipher text continuously. Once a refresh is performed group controllers stop forwarding JOIN and LEAVE messages to other group controllers. Senders and receivers continue to send and receive as they normally would. They join the CGC nearest to them in the network. Thus there is a hierarchy of subgroups. A DGC which does not function as per the group requirements may be black listed and barred from functioning as a DGC after receiving feedback from group participants.

The root of the distribution tree encrypts the data using the algorithm and the receivers decrypt the data using the algorithm. When a transformation is applied by an intermediate node, the receiving nodes are given the proxy encryption key and the intermediate node applies the transformation to modify the cipher text using the algorithm. The receivers can now decrypt the data using the decryption algorithm with the proxy decryption key. The success of the key management scheme thus depends on the ability of the group controllers to distribute the proxy decryption key to the required participants and the proxy encryption key to the required intermediate nodes. The concept of cooperative key generation for generating the session keys is used for exchanging messages between two nodes as can be seen in the case of communication between a CGC and the group participant in section 5.4. RSA, Diffie-

Hellman and elliptic curves are asymmetric key technologies to exchange information between two nodes and generate a single cryptographic key. Any of these key technologies could be used.

5.4 OPERATIONAL ENTITIES

The various entities and their operational details are:

5.4.1 GROUP MANAGER

A node is configured as the Group Manager or the initiator and it is the responsibility of this node to create or obtain from the network administrator a multicast group access control list. This access control list would contain identities of potential members of the multicast group. Only these members will be allowed to join the multicast group. This access control list can be digitally signed like an X.509 certificate [24] so that it can be authenticated when given to the CGCs. The CGC can verify the digital signature to determine if the access list has been modified during transmission. The Group Manager will also have a list of CGCs called the control group controller list (CGCL) so that only trusted nodes will have the capability to allow members to join the network or to distribute keys. The CGCs in the list will also have priorities assigned to them. These priorities will determine the preference set by the network administrators as to which CGC should become the new Group Manager in case the current Group Manager fails. The priorities can be a single digit with 0 indicating the highest priority. The default priority if there is no priority assigned is 9. If there are a number of CGCs with equal priorities to choose from then the CGC with

the lowest IP address is selected. The Group Manager also maintains an exclude group controller list (EGCL). These are those CGCs who do not wish to receive messages because they do not have any group members in the vicinity.

When a node is configured as or assumes the role of Group Manager, it sends a Hello message to the CGCs with the access control list, the CGCL, timers and required key information. These Hello messages could be further extended to include other management information such as group permissions, compromise lists of erring hosts, re-key intervals depending on the features of the group, etc. The Hello messages may be sent using a TCP connection or a reliable multicast protocol connection to ensure that they are not lost. The group manager sends Hello messages to all CGCs excluding those in the EGCL periodically to inform them of any changes in information, to send them re-keying information and to inform them that it is alive and working. These subsequent Hello messages may contain updates about the access control list, the CGCL or required key information if there are any changes. Ideally only the first Hello message will contain information for group management. The remaining Hello messages would only contain re-keying information unless there are any changes. Hello messages are sent by the group manager every Hello_Period. The periodicity of these messages will be kept quite large as the scheme can ensure localized key changes without frequent re-keying. The probability of active nodes capable of functioning as Group Manager's going down frequently in a network is also low. These messages are repeated after a large period or when there are changes or when there is a re-key. Thus the overhead caused by these messages is extremely low. A Group Manager may receive a Hello_Request from a CGC to check if the Group Manager is alive and working. If this Hello_Request is from a member in the

CGCL then the Group Manager immediately responds with a Hello message otherwise it ignores the message. The Hello message sent in response does not contain any information about the access control list, the CGCL or any key information. If the Hello_Request is from a member in the EGCL then the Group Manager removes the member from the EGCL and sends it a Hello message with the access control list, the CGCL and the key information. On receiving an Exclude_Me message from a CGC, the Group Manager includes the CGC in the EGCL and stops sending it Hello messages.

5.4.2 CONTROL GROUP CONTROLLER (CGC)

A node assumes the responsibility of a CGC on receiving a Hello message with an access control list and a CGCL. It stores the Group Manager's address, the access control list, the CGCL and the key information. It sets a timer called the Hello_Timer, a timer called the Change_Time, initializes a variable Re_Election to 0 and another variable Need_Exclusion to 0. On receiving Hello messages from the Group Manager, the CGC resets the Hello_Timer and the Re_Election variable. If it receives a Hello message and has no group participants in its vicinity who are corresponding with it then the CGC increments the Need_Exclusion variable by 1. If it receives a Hello message and has participants in its vicinity then it resets the Need_Exclusion variable to 0. If the Hello message contains any changes in the access control list or the CGCL, the CGC updates the stored values.

If the CGC does not receive a Hello message within the time the Hello_Timer expires, it sends a Hello_Request to the Group Manager, resets the Hello_Timer and

increments the Re_Election variable by 1. If the Re_Election variable reaches a certain threshold (say 3, for example), the CGC assumes that the Group Manager is not working and it goes into Election Mode. It searches the list for the next node that can act as a Group Manager depending upon the assigned priorities. It sets the Re_Election variable to 1, resets the Hello_Timer and sends a Hello_Request to the potential Group Manager. If the Hello_Timer expires without any response from the new potential Group Manager, the CGC resets the Hello_Timer, increments the Re_Election variable by 1 and sends a Hello_Request again. If it receives a Hello message from the new Group Manager it updates the stored Group Manager's address, resets the Re_Election variable and the Hello_Timer and comes out of Election Mode. If it does not get any response and the Re_Election variable reaches 3, the CGC assumes that the node is not working and repeats the process with the next member in the CGCL. It continues this until it receives a Hello message from a new Group Manager or its own turn comes to become the Group Manager in which case it assumes the role of Group Manager. It then comes out of Election Mode. If a CGC receives a Hello_Request from another CGC and it is in Election Mode it immediately assumes the role of Group Manager and comes out of Election Mode. If a CGC receives a Hello_Request from another node and it is not in Election Mode then it immediately goes into Election Mode. The maximum number of messages sent by each CGC to the Group Manager and the potential new group manager is limited to three and these messages are not duplicated by other CGCs so the problem of uncontrolled flooding will not occur. It should be noted that when a CGC becomes a Group Manager it starts by sending a Hello message to the CGCs with the access control list, the CGCL and required key information. It no longer needs to maintain the variables Re_Election and Need_Exclusion. If the Need_Exclusion variable

reaches a certain threshold (say 9, for example), the CGC knows that it has seen a fixed number of Hello messages without having any participants in its vicinity. It sends an Exclude_Me message to the Group Manager and goes into Dormant Mode. In Dormant Mode the CGC does not expect any Hello messages. The CGC comes out of Dormant Mode when it receives a message from a participant. It then sends a Hello_Request to the Group Manager.

5.4.3 DATA GROUP CONTROLLER (DGC)

Any node in the network can be designated a DGC as they do not have access to keys or to any clear text messages. They cannot derive any actual keys from the information they possess. They are only used for converting cipher texts and not for authenticating group participants or exchanging encryption or decryption keys.

5.5 OPERATIONAL OVERVIEW

A detailed operational overview by considering each possible scenario in a multicast group life cycle is as follows:

5.5.1 GROUP CREATION

Group creation is a three-step process that involves initializing the group, creation of the group keys and distribution of the group keys to the CGCs. This process may overlap with group members starting to join the group. Initializing the group is the process of configuring a node as the Group Manager and providing it with access lists

and a list of other trusted nodes in the network. The group parameters may be published using a directory service. The Group Manager creates the encryption and decryption keys. The Group Manager then sets up connections with the CGCs. It will set up a session key encryption key (SKEK) to secure any communication between itself and the CGCs in various parts of the network. It then uses this connection to send Hello messages to the CGCs. Though it is not necessary, having a number of CGCs in different parts of the network would improve the scalability. Any node in the network can be assigned a DGC by any CGC unless the node is unsuitable for some reason.

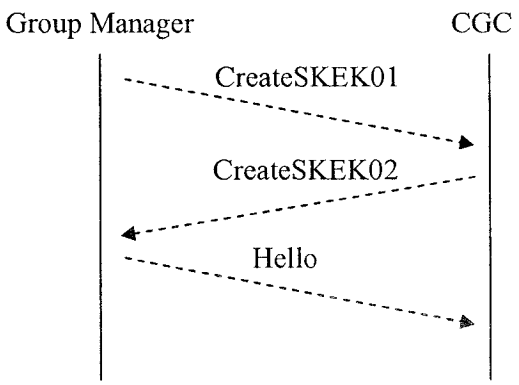

Figure 12: Messages during Set-up

Figure 12 shows the various messages between Group Manager and the CGC. The messages CreateSKEK01 and CreateSKEK02 contain the random values necessary for generating the SKEK and the public keys of the respective entities. The messages are validated and the SKEKs generated and stored. The public keys are also stored. The Group Manager then sends the Hello message to the CGC, which contains the necessary information for management of the group. If a reliable multicast protocol is being used to send Hello messages then the Group Manager can send the key used to

encrypt the Hello message to all CGCs using the SKEK to the new CGC and then multicast the Hello message.

5.5.2 MEMBER JOIN

To join a secure multicast group, a participant must request the decryption key from a CGC, authenticate itself and then receive the key. The participant sends a JOIN message to its closest Group Controller using a secure unicast channel. On receiving this request the group controller decides whether to approve or deny the request. The group controller shares a SKEK with the group member to protect communications between itself and the participant. This is an intrinsic behaviour that is required of all authorization procedures over the Internet and not a function added here.

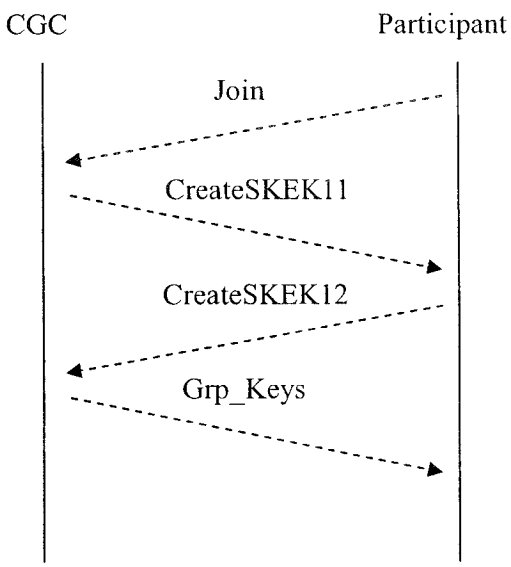

Figure 13: Messages during Member Join

Figure 13 shows the various messages that will be exchanged during a member join. The Join message is a membership request to join the group sent by the participant to

the CGC. The next two messages are for establishment of the SKEK. CreateSKEK11 contains the random value necessary for the participant to generate the SKEK. The participant validates the message, generates the SKEK and stores it. It sends a message CreateSKEK12 which contains the random value necessary for the CGC to generate the SKEK. It also contains necessary information encrypted using the SKEK that identifies the participant as having permissions to join the group. This could be information that can be verified by the CGC using its access control list. The CGC validates the message and generates the SKEK.

If the group member is a valid key member and the timer Change_Time has not expired then the CGC gives the decryption key to the group member thereby allowing it to join the group. Details of the timer Change_Time are explained in section 5.5.6. If the group member is a valid member and the timer Change_Time has expired then the CGC splits the decryption key to the proxy encryption key and the proxy decryption key. It sends the proxy decryption key to the participant over a secure channel using the SKEK, which it shares with this participant. This message is Grp_Keys. If the group controller is a DGC, it passes the message to the CGC nearest to it who has an access list. The CGC splits the key to the proxy encryption key and the proxy decryption key. It then informs the DGC if the participant has access to the group by sending it the proxy encryption key and sends the proxy decryption key to the participant using a secure channel by generating and sharing a key SKEK with it. If the participant has access to the group then the CGC also sends a message to the DGC containing the proxy decryption key encrypted using the decryption key.

The remaining group members under CGC or the DGC are notified of the change in the key by multicasting the proxy decryption key to them encrypted using the current

key decryption key. They now use the proxy decryption key as the decryption key. The group controller applies the proxy transformation under the proxy encryption key to the multicast data now arriving before forwarding it to the participants. Participants can now decipher the encrypted and transformed multicast data using the proxy decryption key. As the new group member does not know the previous decryption key it cannot access the past data ensuring perfect backward secrecy. DGCs have access to only the proxy decryption key and are unable to guess the decryption key. Thus they cannot compromise the key.

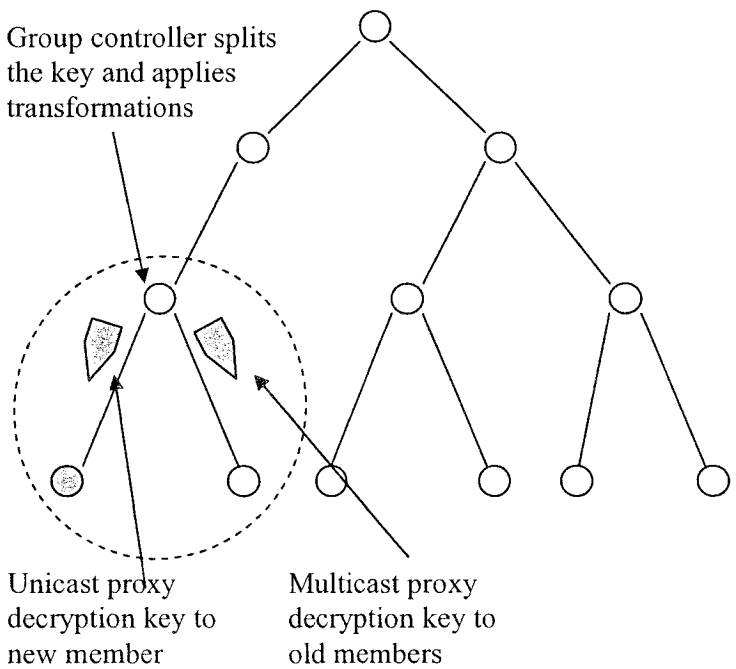

Figure 14: Member Join

Figure 14 shows the case when a new receiver joins the distribution tree. The group controller splits the key and gives the proxy decryption key to the participants and starts transforming the data using the proxy encryption key.

5.5.3 MEMBER LEAVE

A participant may leave a multicast group silently by not refreshing the JOIN message. Each JOIN message thus has a timer associated with it. If a participant wishes to remain in the group it must refresh the JOIN state using a secure channel. Thus each group controller has a soft-state associated with it. Once the state expires the key is changed as described below and the participant is excluded from the multicast group. The participant may voluntarily leave the group by sending a leave message. If the group policy requires perfect forward secrecy then the keys are changed and the participant is excluded. In certain cases it might be necessary to forcibly exclude a hostile participant. A hostile participant may be added to the list of erring hosts which can be sent using Hello messages to prevent future access of this participant.

If a CGC receives a LEAVE message or decides to exclude the participant due to a JOIN state not being refreshed and the timer Change_Time has expired then it splits the decryption key to the proxy encryption key and the proxy decryption key. It sends the proxy decryption key to the remaining participants under it. One way to do this would be to unicast it over separate secure channels to each participant. Another way would be to encrypt the proxy decryption key using the KEK for each group member and multicast one message. This message contains the same proxy decryption key encrypted differently for each group member. Thus a single longer message replaces separate unicast messages. The group controller uses the proxy encryption function with the proxy encryption key to change the cipher text.

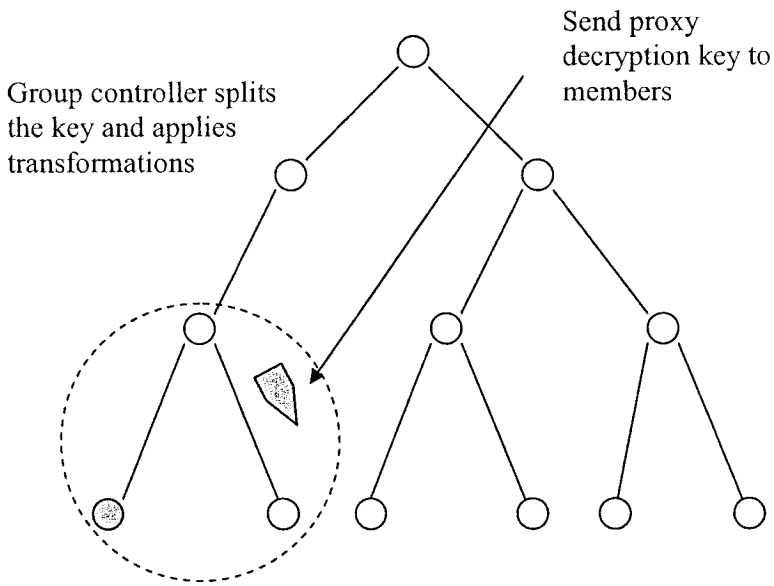

Figure 15: Member Leave

If a CGC receives a leave message or decides to exclude the participant due to a JOIN state not being refreshed and the timer Change_Time has not expired then it waits for the timer to expire before splitting the key and proceeding further. If a DGC receives the LEAVE message, it forwards the LEAVE message to a CGC near it. The CGC then splits the decryption key to the proxy encryption key and the proxy decryption key. The proxy decryption key is sent to the remaining participants under the DGC using the same mechanism as for the CGC directly receiving the LEAVE message. The proxy encryption key is given to the DGC. The DGC applies the proxy transformation function. The participants can access the encrypted and transformed multicast data using the proxy decryption key. As the participant who has left the group does not have the proxy decryption key, it cannot access the multicast data, thus ensuring perfect forward secrecy.

Figure 15 shows the case when a receiver leaves a multicast group. The group controller splits the decryption key and gives the remaining participants the proxy decryption key. It then starts transforming the multicast data using the proxy encryption key. If there are a number of CGCs on a data path from a sender to a receiver then the transformation must take place only once along the path. Any CGC when it receives a message notifying a change in the key must stop performing transformations itself. This ensures that the transformation takes place only once and as it is taking place higher up in the tree, the CGCs lower down in the tree do not need to perform any transformation.

5.5.4 DATA TRANSMISSION

The sender can directly encrypt the data using the encryption key given to it by the group manager. Since the decryption is being performed by the participants directly there is no need for the sender to bother about the data transmission after encryption. Group controllers who are required to change the cipher text are able to do it without seeing the clear text message and they too need to decrypt only once. Group controllers perform the transformations only when required.

5.5.5 RE-KEYING

The group manager may periodically refresh the key by changing the encryption key and the decryption key. The group manager informs the senders to use the new encryption key to encrypt the data. It sends the new decryption key to all the participants by encrypting it using the old encryption key and sending it using a single

multicast. It also sends the key to all the CGCs by sending the new key to them in a Hello message. The CGCs on receiving the new key stop transforming the multicast data thereafter as the key is changed and there is no need to transform the data anymore. If the CGCs had asked any DGCs to perform the transform they are asked to stop as well.

5.5.6 BATCHING

The concept of batching is used to ensure that the splitting of keys and transformations are not performed repeatedly unnecessarily. SIM-KM performs transformations only after a certain amount of time has passed. The length of this time period is decided by the Group Manager depending upon the nature of the multicast data and the requirements of the group. This information is broadcast to the CGCs using the Hello message. This time period, known as Change_Time, may be changed during the lifetime of the multicast group. It may be kept large during the start and end periods of the multicast group as a lot of participants may join and leave the multicast group during this time. If it is not very sensitive data then there is no point in continuously changing the keys. This time period is reduced as the group becomes stable and then the keys are changed on most membership changes. This ensures that the splitting of keys and transformations do not occur repeatedly. Thus the join and leaves received during the time interval that the timer is running get aggregated. This reduces the frequency and hence the cost of changing keys and performing transformations. Cumulative members join and leave improves the functioning of the protocol.

5.5.7 SHUTDOWN

A shutdown basically implies the group being shutdown by the group manager. This could happen by the soft states being deleted as they are not refreshed or multicast data not being received and the group state in the nodes expiring as no more senders are sending data. The group manager may explicitly decide to shut down the secure group by informing the group controllers not to perform the transformations or forwarding anymore.

5.6 ANALYSIS

Proxy encryption schemes have been shown to be as secure as the original schemes [23]. Hence the levels of difficulty of being able to decrypt and obtain the data are the same as that of the encryption scheme used. This section presents an analysis of possible attacks. It also compares SIM-KM with other group key management schemes.

5.6.1 VALIDATION

We have used PROMELA (PROcess MEta LAnguage) [25] to specify the validation model and then used a tool, SPIN (Simple Promela INterpreter) [26] to validate our model. The model is designed so that it is simple but considers the different tasks involved in SIM-KM. The model consists of a simple hierarchy of nodes as shown in figure 16.

The Group Manager (GM) distributes keys at the beginning of the multicast group. The receivers are made to join and leave the multicast group and the functioning of the CGCs and DGC are observed. The sender sends data packets.

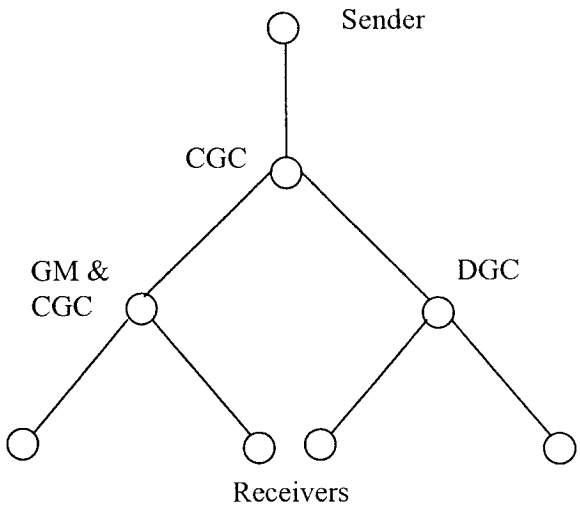

Figure 16: SIM-KM Test Model

XSPIN is used to specify the high level model written in PROMELA. As a preliminary check different random simulations were performed with different SPIN options and no errors were found. The verifier was compiled using the exhaustive search option. This option causes a state space search of all possible states and message timings of the modelled processes.

Then, the verifier was executed and the output confirmed that our model is free from errors and there are no assertion violations, invalid end states or unreachable states in the design. The working of the CGCs and DGCs was observed and they were found to function as expected.

5.6.2 COMPARISON WITH OTHER SCHEMES

Table 1: Comparison of Group Key Management Schemes

Characteristic	GKMP	SKMD	Iolus	SIM-KM
Scalable	No	No	Yes	Yes
Trust in third parties	No	Yes	Yes	Partial
Compromise recovery	No	No	No	Yes
Single point of failure	Yes	Yes	Yes	No
Bandwidth overhead to accommodate PFS/PBS	Very Large	Very Large	Small	Small
Delay at intermediate nodes without PFS/BFS	None	None	Large	None
Delay at intermediate nodes with PFS/BFS	None	None	Large	Minimal

As seen from Table 1 the centralized schemes are not scalable as they cannot handle membership changes for large groups. There is also a very high bandwidth requirement if they are to accommodate perfect forward secrecy (PFS) and perfect backward secrecy (PBS). The distributed subgroup schemes are able to handle subgroup changes much better than the centralized schemes. However they suffer from delays at intermediate nodes caused by the processing time required to decrypt and re-encrypt packets. Even Mykil, which is a mixed scheme, suffers from this drawback. Other schemes such as LKH require the intermediate nodes to store a large number of keys. They are also complex as they require specific keys to be distributed to specific nodes. SIM-KM fits all the requirements. It performs as a fault tolerant centralized scheme when PFS and BFS are not required. As the control group controllers are in various parts of the network they reduce the time required to respond to membership requests by participants. They can also take over functioning

of the group if the group controller fails. The data group controllers can transform the data when required. They are located near the edges of the distribution tree to reduce the effect of group membership changes. It has minimal bandwidth overhead and requires minimal processing at intermediate nodes to support PFS and BFS.

5.6.3 MSEC INTEGRATION

The proposed scheme is a seamless extension to the group key management architecture defined given in [10].

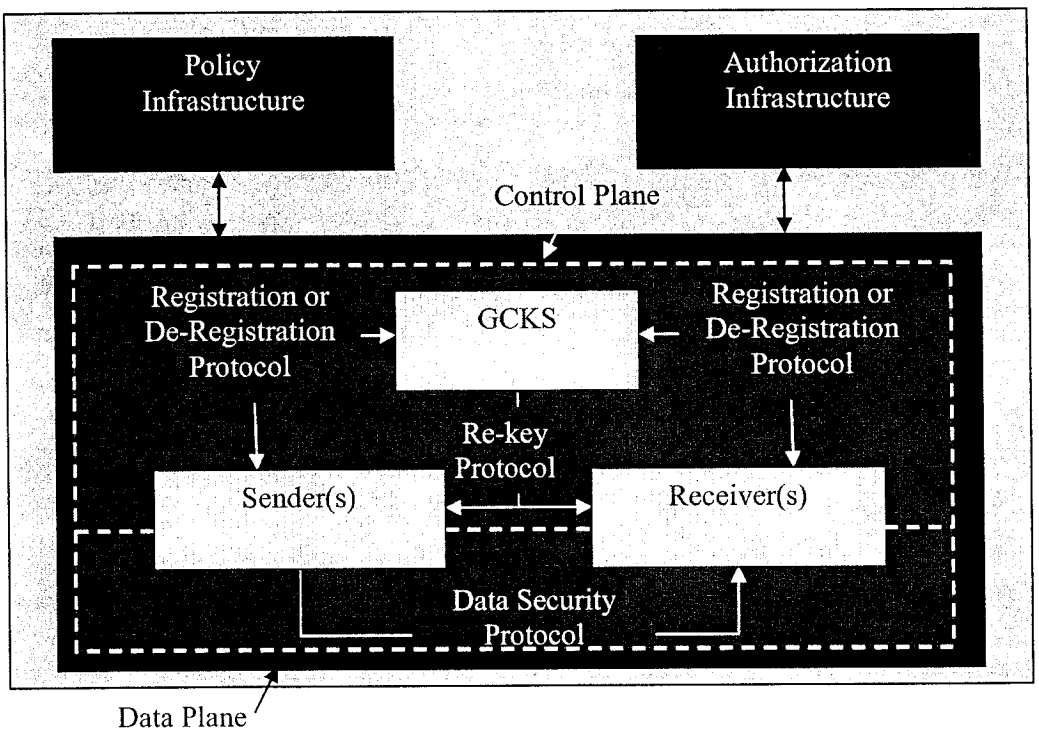

Figure 17: Group Security Architecture Map

Figure 17 shows the group security architecture and how the proposed scheme fits in with the architecture. It also shows how an extension of the control plane would be able to register and de-register group members, perform the re-keying operation and

interact with the policy and authorization infrastructure to apply multicast security policies and administrative operations. The data plane handles the flow of data between senders and receivers. Group controllers having access lists may be determined by the network administrator or trusted group controllers may obtain access lists from other trusted group controllers. Group controllers performing only the cipher text modification may be chosen dynamically by checking if the fork in the multicast distribution tree has more branches than one.

5.6.4 PERFORMANCE ANALYSIS

Any encryption scheme may be used for protecting the data. For the purpose of comparison of various schemes in our simulations we have used RSA [27] as the encryption scheme. It must be noted that other schemes that are faster are available and other implementations that use certain techniques for speeding up the calculations are available. Some of these schemes may be implemented in hardware and will actually improve the performance of all the schemes substantially. However this simulation compares relative speeds of the group management schemes and the bandwidth usage clearly demonstrating which scheme is better when a certain encryption scheme is chosen.

ENCRYPTION/DECRYPTION/TRANSFORMATION:

To find out the time taken to encrypt, decrypt and transform data by the same machine we implemented the RSA algorithm. Key generation and transformation can be performed simultaneously with data distribution or could be pre-computed so it does not affect the transmission time. The key size was kept 128 bits. The size of the key

required depends on the value and nature of the content it protects. It makes no sense for an adversary to spend (say) $10 million breaking a key if recovering the key will only net (say) $10 thousand [28]. The key is also changed periodically so to decrypt the entire content would be much more difficult. Session keys and keys for exchange of control information will be kept much larger as per prescribed guidelines for key sizes [29]. Key sizes for protecting multicast data can be kept small as the value of the content they protect is much smaller. They may be kept larger for certain applications where sensitive data are being broadcast.

The RSA algorithm was implemented in C++ using Visual Studio. The programs were run on a Pentium IV, 1.99 GHz, 256 MB RAM machine running Windows XP Professional Edition. The algorithm was run 100 times and the worst case performance was taken to show the maximum delay that would occur due these algorithms.

Table 2: Performance of RSA algorithm

Task Performed	**Number of Clock Ticks**	**Speed (Mbps)**
Encryption	19	1.507177
Transformation	23	1.245060
Decryption	46	0.622530

Table 2 shows the time taken to perform the various operations. The time taken to split the key into the proxy encryption and decryption key ranges from 6 clock ticks (0.000002 sec.) to 1540234 clock ticks (0.430288 sec.). This does not affect the transmission as the key can be split in advance and stored. It can be used later when a group membership change occurs. It does not affect the time taken to distribute data but the results shows that the time taken to perform the task is small. We use these

results for calculating the transmission times and bandwidth used by the different group key management schemes.

GENERATING NETWORK TOPOLOGIES:

Under the hierarchical model [30], the Internet can be viewed as a collection of interconnected routing domains, which are group of nodes that are under a common administration and share routing information. Each routing domain in the Internet can be classified as either a stub domain or a transit domain. A domain is a stub domain if the paths connecting any nodes u and v go through that domain only if u or v is in that domain; transit domains do not have this restriction. The purpose of transit domains is to interconnect stub domains efficiently; without them, every pair of stub domains would need to be directly connected. The hierarchical graph required for the simulation is generated from the hierarchical network graphs available in network simulator (ns2) [31]. The hierarchical graphs in ns2 have been generated using the transit-stub model described above using Georgia Tech's SGB (Stanford GraphBase) graphs. Nodes in the graphs are designated to work as senders, receivers or group controllers depending upon the multicast scenarios. The choice of senders, receivers and group controllers is kept the same for all the simulations for a particular scenario.

SIMULATION:

For the purpose of the simulation application areas where multicast is a potential asset were selected and for each of those cases separate simulations were executed [32]. In order to anticipate what the most common multicast applications will be, we need to examine the reason why one would use multicast. The reason for using multicast is to overcome inefficient use of network resources such as: bandwidth, transmission time

and computational resources. This becomes relevant when there is more than one participant in the session. The larger the number of participants, the larger the ability to save and the larger the volume of traffic, the more the crucial the problem becomes, and thus the savings are more predominant. The following is a list of multicast application scenarios:

- Shared whiteboards
- Multi-party audiovisual conferencing
- Multimedia streaming/Pay per view
- Data distribution

Table 3: Group Members in Each Scenario

Scenario	**No. of senders**	**No. of receivers**
Shared Whiteboard	7	7
Multi-party Audiovisual	14	14
Multimedia Streaming	1	65
Data Distribution	1	98

The simulation has been performed using Network Simulator – ns2 [31]. NS is a discrete event simulator targeted at networking research. The simulation is viewed using Network Animator (NAM). NAM is a Tcl/TK-based animation tool for viewing network simulation traces and real world packet traces. The various group key management schemes were modelled in ns2 for running simulations. For all the simulations we consider the network links to be bidirectional with a bandwidth of 10 Mbps and delays ranging from 100 to 500 ms chosen at random. The simulation builds the network according to the hierarchical model. Some nodes in the network are designated to work as group controllers. These nodes function as group controllers

in the schemes that require distributed group controllers. The nodes functioning as group controllers were kept same for all the schemes.

The underlying multicast routing protocol used is Protocol Independent Multicast – Sparse Mode (PIM-SM) [5]. It makes sources and receivers according to the scenario. The membership dynamics of the various groups were modelled depending on the scenario and from actual data [33]. The number of senders and receivers for each of the scenarios is shown in Table 3. In all the scenarios we assume that there is a need for perfect forward and backward secrecy and that the key must be changed at every membership change.

Shared whiteboards are software applications that allow users to participate in distributed meetings. The data transmitted are small and intermittent. Group membership is not very dynamic. Figure 18 shows the comparison of bandwidth usage in the network over time for a shared whiteboard application. The performance of the schemes is comparable as the number of senders and receivers is low, the group membership is not dynamic and the data rate is small. Multi-party audiovisual conferencing applications have highly dynamic group memberships. The data transmitted is high and continuous. For example: The 43rd International Engineering Task Force Committee (IETF) meeting video had a total of 129 receivers with a maximum of 90 receivers at any given time and an average membership of 48.59%. As seen from Figure 19, the bandwidth usage is comparable for SKMD, Iolus and SIM-KM but the time taken by Iolus to deliver the same amount of data are higher. GKMP uses a high amount of bandwidth whenever there is a change in group membership as the keys have to be distributed to all members.

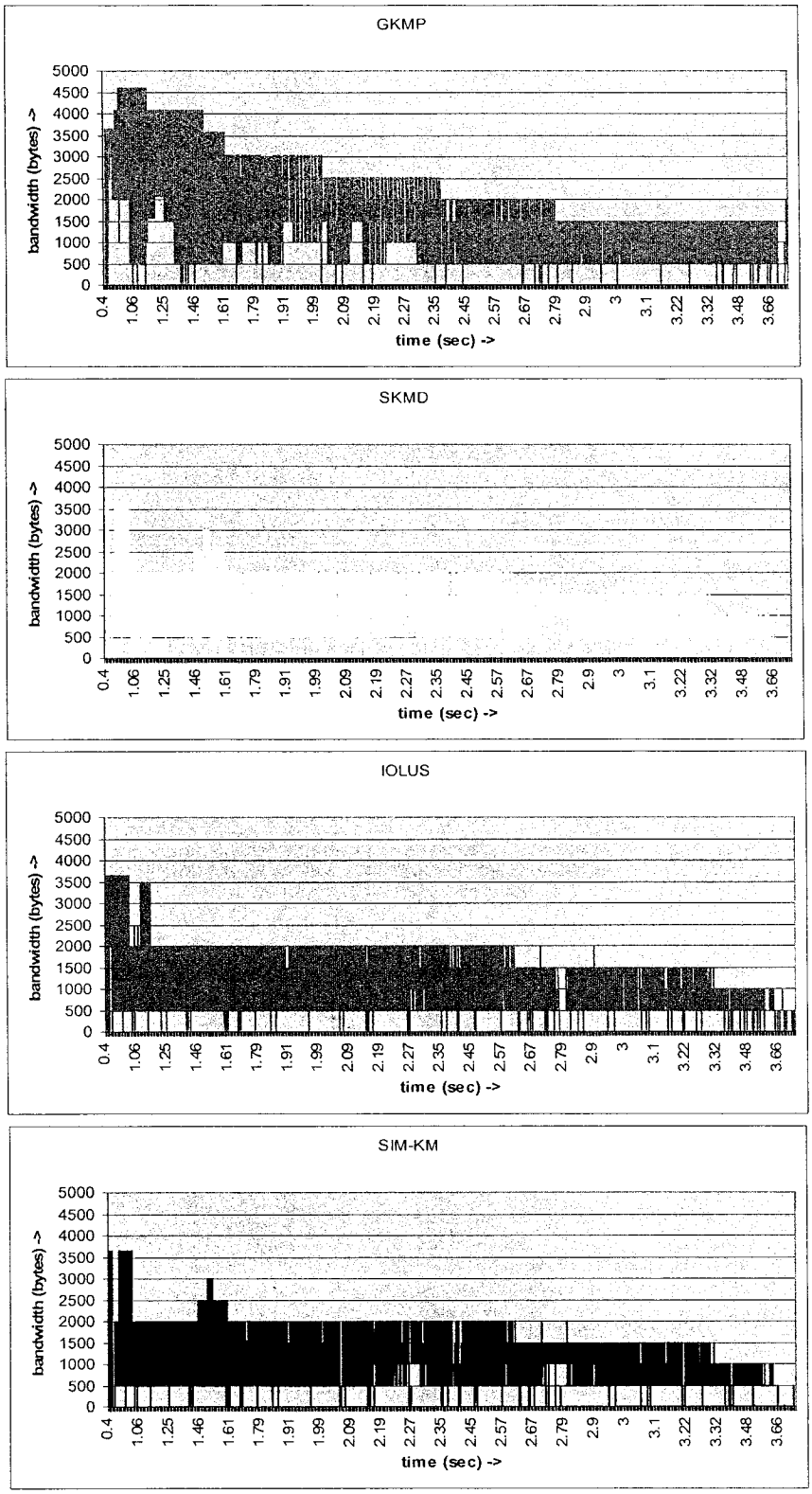

Figure 18: Whiteboard Application

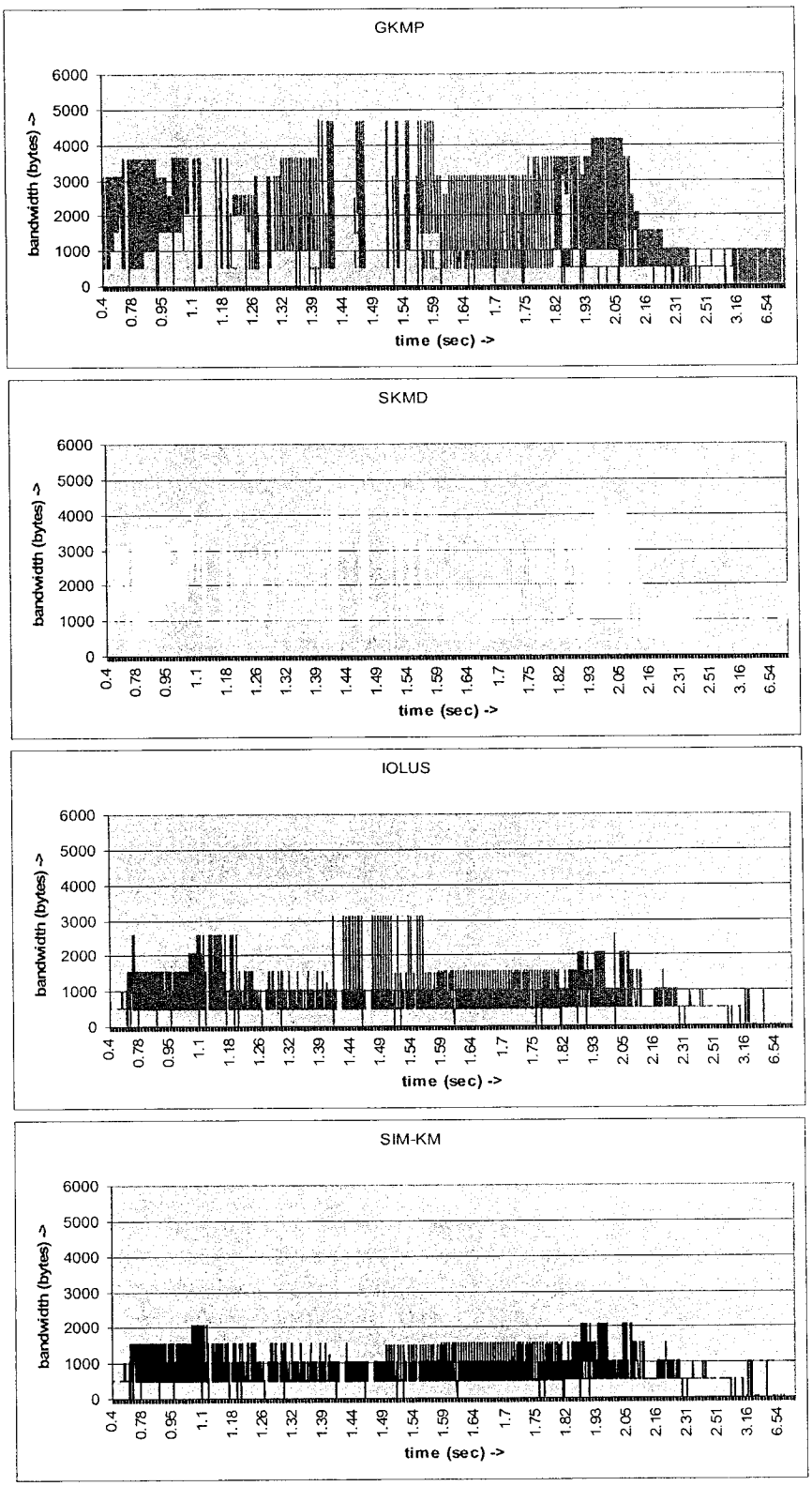

Figure 19: Multiparty Audio Visual Conferencing

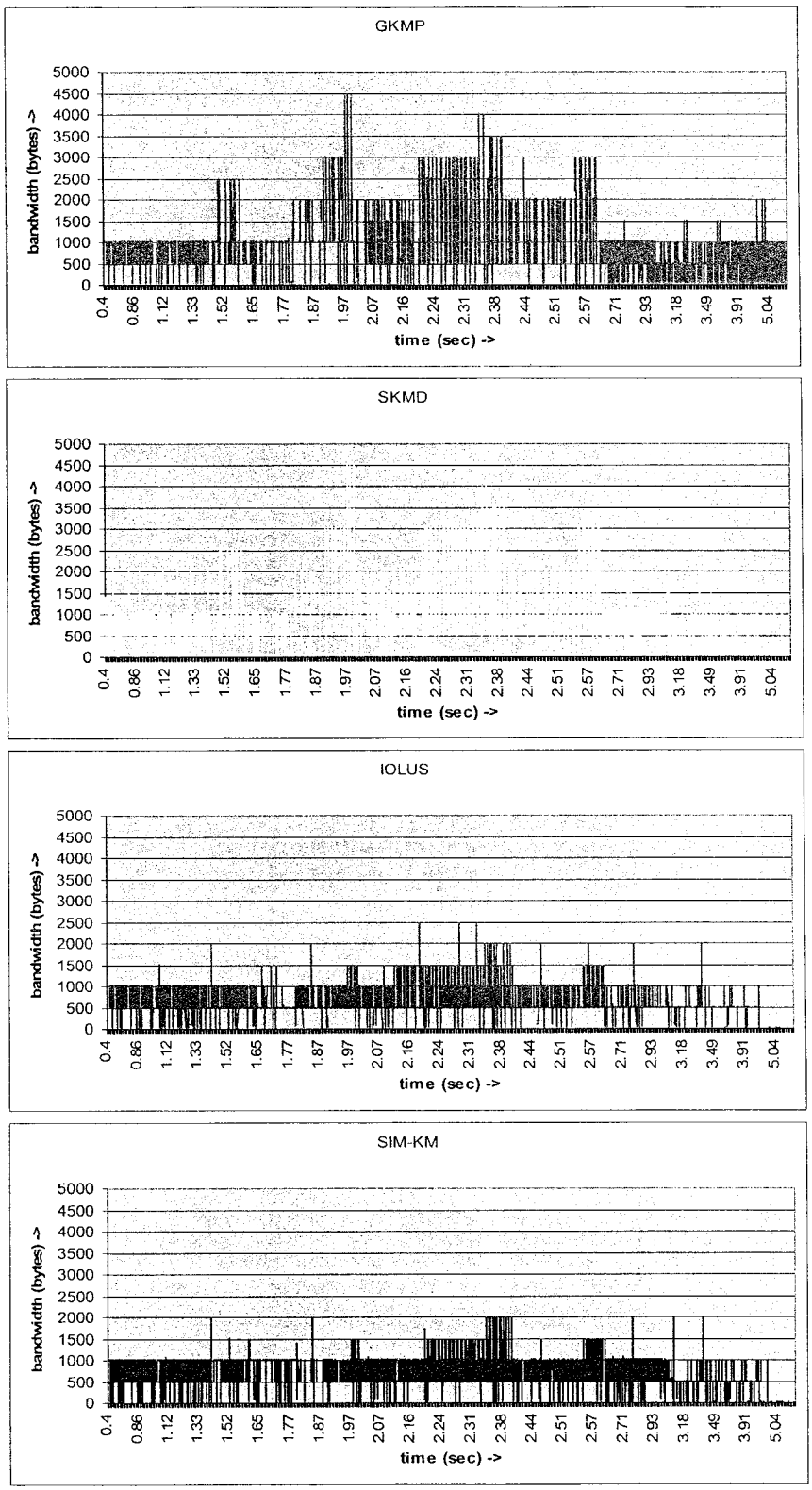

Figure 20: Streaming Multimedia

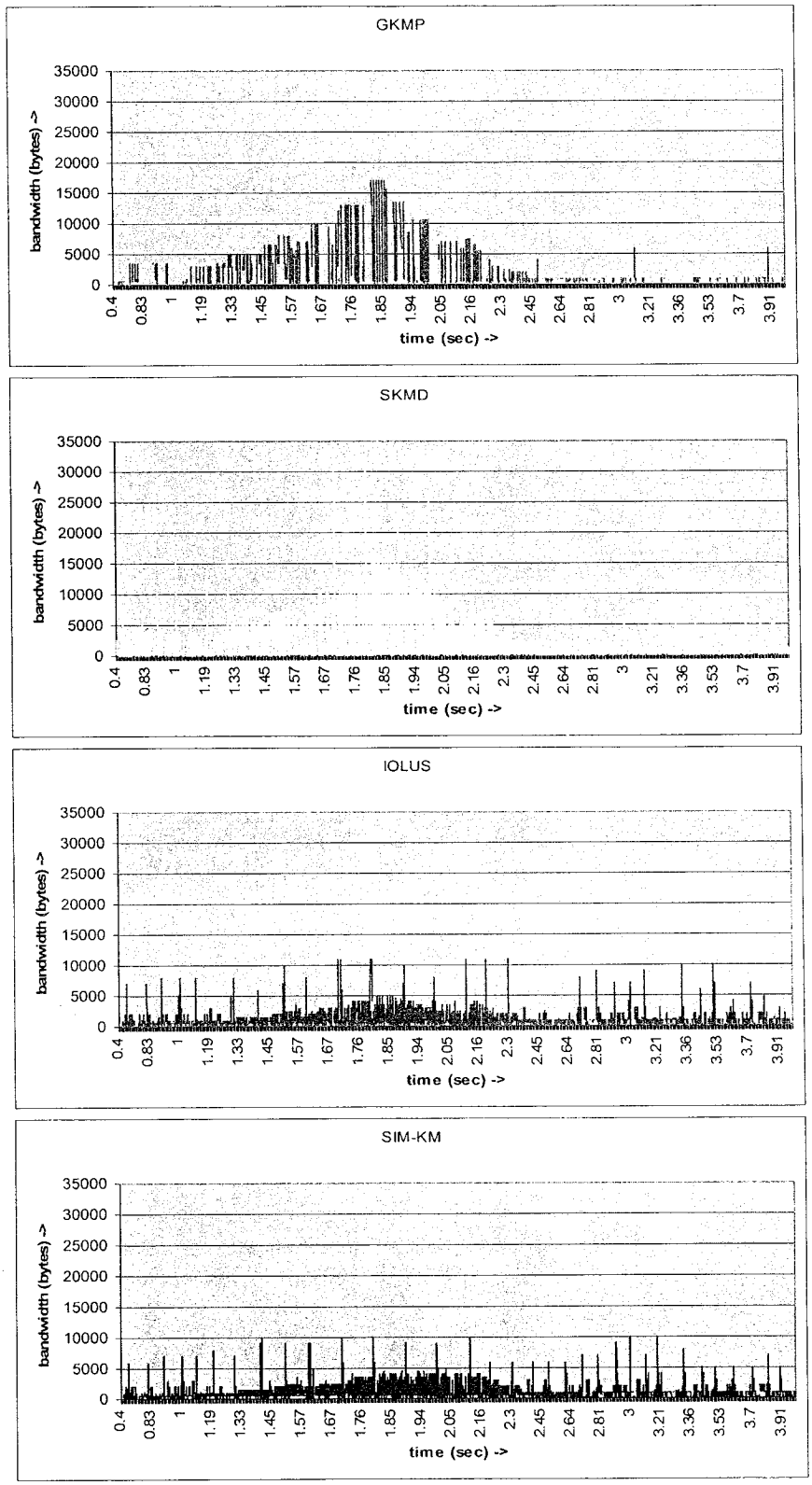

Figure 21: Data Distribution

In streaming multimedia and pay-per-view applications the group membership is dynamic. The data transmitted is high and continuous. For example: The NASA shuttle launch in February 1999 had a total of 62 receivers with a maximum of 62 receivers at any given time and an average membership of 40.33%.

As seen from Figure 20, Iolus uses up a lot of bandwidth because of retransmissions caused due to dropping of packets from the node's queues. The queues are formed due to the time taken to decrypt and re-encrypt the packets. It also takes a longer time compared to the other schemes. In data distribution applications the group membership is not at all dynamic. The data transmitted is high and continuous. As seen from Figure 21, Iolus suffers from the problem of decryption and re-encryption. In all the scenarios SIM-KM performs better than or comparable to the other schemes.

A comparison of the average network bandwidth used by the various schemes is shown in Table 4.

Table 4: Average Bandwidth Consumption

Scenario	GKMP	SKMD	Iolus	SIM-KM
Shared Whiteboard	1389	1071	1053	1071
Multi-party Audiovisual	1292	1402	776	1040
Multimedia Streaming	886	975	630	685
Data Distribution	1569	1252	950	838

-All values are in bytes

The average network bandwidth used by SIM-KM is comparable to the average network bandwidth used by the Iolus scheme and lower than that used by GKMP and SKMD. SIM-KM does not incur any delay and has very low processing overhead at

the nodes compared to Iolus. The performance of SIM-KM over the other schemes improves as the group size increases. There is a sudden increase in network bandwidth usage when group membership changes or when a re-key operation is performed for GKMP and SKMD as seen from the graphs. This sudden increase does not occur in Iolus and SIM-KM.

The simulations were also executed for streaming multimedia application and the average packet delay was measured.

Table 5: Average Packet Delay

Scenario	GKMP	SKMD	Iolus	SIM-KM
Multimedia Streaming	1447	1329	1016	712

-All values are in milliseconds

Table 5 displays the average time taken for a packet to reach from the sender to the receivers. Due to network congestion caused as a result of re-key messages in GKMP the average packet delay is the highest of the four cases. SKMD also suffers from network congestion and hence the average packet delay is high. Iolus has better network bandwidth usage but suffers from having to decrypt and re-encrypt the packets. SIM-KM uses network bandwidth efficiently and also has to perform transformations only at certain times. Hence the average packet delay is much lower when compared to other schemes.

6 DATA SOURCE AUTHENTICATION REQUIREMENTS

The need for multicast authentication was outlined in section 1 and in section 2.3.2. Authentication takes two flavours:

- ***Source authentication and data integrity***: This functionality guarantees that the data originated with the claimed source and was not modified en route (either by a group member or an external attacker).
- ***Group authentication***: This type of authentication only guarantees that the data were generated (or last modified) by some group member. It does not guarantee data integrity unless all group members are trusted.

Source authentication is difficult to achieve. The kind of authentication required depends on the context of the application. The multicast security service must handle authentication and integrity verification of multicast data. It includes the transforms to be made both at the Sender's end and at the Receiver's end. A multicast authentication scheme must have the following properties:

- Low computation overhead for generation and verification of authentication information
- Low communication overhead
- No buffering required for the sender and the receiver, hence timely authentication for each individual packet
- Strong robustness to packet loss
- Scales to a large number of receivers
- Protects receivers against denial of service attacks in certain circumstances if configured appropriately

- There should be no time synchronization required between the Sender and the Receiver.
- The scheme should be secure against a powerful adversary with full control over the network. The adversary can eavesdrop, capture, drop, re-send, delay, and alter packets.

7 TAXONOMY OF AUTHENTICATION APPROACHES IN GROUP COMMUNICATIONS

There have been a few approaches to multicast authentication. In this section we present an overview of some of the current approaches.

One approach is to use MACs where the group members share a secret key and a MAC is included in every packet. In this scheme any group member can spoof packets. To avoid this each receiver can be given a secret key and the sender can have all such keys. The sender now has to add a MAC for each receiver. This scheme is prone to collusion attacks and the size of each packet increases with the number of receivers resulting in enormous communication overhead. The Multiple MACs scheme [34] is not scalable and suffers from large communication overhead. Timed Efficient Stream Loss-tolerant Authentication (TESLA) [35] has low communication overhead but requires time synchronization between senders and receivers, which is difficult to maintain in large groups.

Another approach is to use digital signatures. As this is a computationally intensive operation, schemes work with fast signature techniques and amortize a signature operation over several packets [36] but this scheme does not tolerate packet loss. BiBa [37] is a fast individual packet authentication signature scheme but requires time synchronization between sender and receivers, which limits the authentication rate and also suffers from communication overhead. A Merkle hash tree [38] can be used for authentication and the scheme is tolerant to packet loss but has enormous communication overhead. Erasure codes [39] can also be used for authentication. The

scheme performs encoding twice to reduce communication overhead but the scheme fails if a single packet is injected. Graph-based authentication schemes amortize a signature over a hash chain in such a way so as to tolerate packet loss. Hashes can be inserted in strategic locations to make it resistant to a burst loss [40]. In general, graph-based schemes offer probabilistic security guarantee. They do not consider adversarial packet losses caused by attackers. The RSA Digital Signature Algorithm can be used for authentication [41]. This scheme is prone to replay attacks and requires use of the RSA signature algorithm, which is very costly in terms of processing time.

The schemes for multicast authentication present today suffer from one of these drawbacks:

- They are computationally very expensive hence they cannot be used with a wide variety of devices. With the widespread use of PDAs, wireless devices, Internet enabled mobile phones and other low power devices with limited resources it becomes infeasible to use computationally intensive techniques for the kinds of applications multicast is targeting such as stock updates.
- They introduce a large overhead communication overhead. With the ever increasing use of audio and video streams bandwidth is limited. Wireless networks also have limited bandwidth. It is unacceptable to clog the network with enormous communication overhead.
- They require time synchronization between the sender and the receivers. With the receivers located at different locations for applications such as online stock quotes, pay-per-view TV, etc., it becomes difficult to maintain time synchronization between the sender and the receivers. Wireless devices are

mobile and their distances changes, which changes the network propagation delay frequently. This requires constant resynchronization of the sender and the receivers making it impossible to maintain the service efficiently.

- They are prone to collusion attacks. While senders are few (in most cases it is only a single sender group) and are highly trusted, the receivers cannot be trusted not to form collusions and share keys. Hence it is unreasonable to accept a solution where receivers can spoof one another.

None of these schemes uses the fact that multicast data for commercial purposes will be protected such as pay-per-view TV, etc., so that people without access cannot see the multicast traffic. This knowledge can be used to add a simple robust multicast packet authentication scheme to any group communication scheme using asymmetric keys.

8 GROUP AUTHENTICATION

In this section we present a technique for performing multicast packet authentication and a message integrity check when asymmetric keys are used to protect the data [42]. As an example, we will show how our technique can provide group authentication when used with SIM-KM. Our method may also be used independently with any asymmetric key distribution scheme. It uses symmetric message authentication codes (MACs) to add data source authentication and data integrity check to secure group communication.

SIM-KM uses asymmetric encryption in a novel method by keeping the encryption key secret and distributing the decryption key to a set of receivers. The use of an asymmetric encryption key makes it possible to perform data authentication by combining the use of a symmetric MAC.

8.1 EXTENDING SIM-KM TO PERFORM GROUP AUTHENTICATION

To provide authentication, a symmetric key is shared among all group members. This symmetric key is a unique shared secret used for authentication. Every time a sender wants to send a message to the group, it adds an index to the packet, a counter "c" and a random number "k". The index is a number assigned by the Group Manager to a particular sender for uniquely identifying it during a multicast session. It then encrypts the packet with the asymmetric encryption key. It then calculates a MAC

(Message Authentication Code) on the cipher text using the symmetric key. It then attaches "k" and the MAC to the packet.

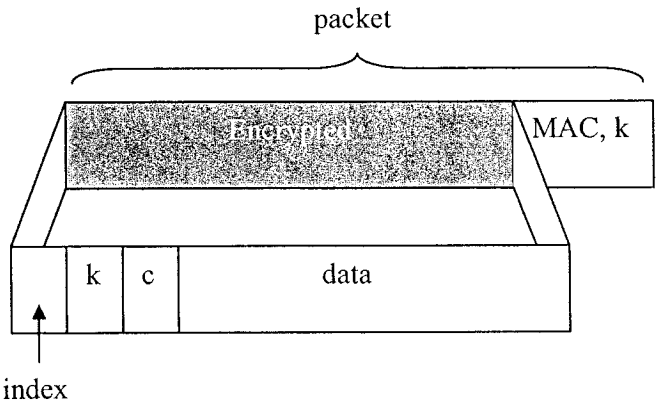

Figure 22: Packet Structure

The packet structure is shown in figure 22. The receiver on receiving the packet computes a MAC to ensure that the packet was not modified in transit and was not sent by someone imitating to be the sender. This however does not rule out the possibility of one of the malicious receivers masquerading as a sender. The packet is then decrypted and the value of "k" obtained after decryption is matched with the value of "k" sent in clear text with the packet. As the packet is encrypted by an asymmetric key, which is unknown to the receivers, there is no possibility that a malicious receiver can create a packet which when decoded produces the same value of "k" without knowing the encryption key. This effectively rules out the possibility of any receiver being able to impersonate a legitimate sender. The index identifies the particular sender providing data source authentication. The receiver stores the last value of the counter received in sequence as c1 and compares it with the value of the counter in the next packet that arrives, to prevent a replay attack. The newly arrived packet must fall within the range c1 to c1 + r, where r is the size of the buffer used by

the underlying protocol to re-sequence the out of order packets. If the packet already exists in the buffer then the packet is considered a duplicate and is discarded.

The symmetric key is generated by the Group Manager and distributed to the group members when they join the multicast group. It may be given along with the decryption key when the members join. The scheme uses a single MAC for message authentication along with a random number, a counter and an index. This reduces the bandwidth overhead considerably and it is not as computationally intensive as digitally signing individual packets. The type of MAC and the size of the symmetric key can be chosen depending upon the context of the application and the sensitivity of the data.

When SIM-KM is used as the key management scheme, the symmetric key is also given to the group controllers. When group controllers perform the data transformation they re-compute and replace the existing MAC. The random number "k" is kept as it is. The decryption key is changed by SIM-KM and is communicated to the receivers who need the new key. The symmetric key for authentication remains the same. If another key distribution scheme is being used then there is no need to re-compute and replace the MAC.

8.2 ATTACKS

This scheme provides message integrity as it allows the receiver to verify that the message is exactly the same as when the sender sent it. Host authentication is also achieved as it allows the sender to be uniquely identified. A number of different

attacks [43] are possible against group communications. We describe some of the attacks that are relevant to message authentication and how the proposed scheme would handle these attacks.

REPLAY:

An adversary may store messages and then send them at a later time. As these packets have been assembled by a legitimate sender, the receivers may be led to believe that they are legitimate packets even though now they are out of sequence. This attack is nullified by having a counter. If packets are replayed then the value of the counter in the packets will be out of the range defined for c and these packets can be discarded.

MESSAGE MODIFICATION:

An adversary may modify messages that are sent by the sender. The symmetric MAC authentication will fail if the message is modified. As only group members have the symmetric key a non-group member cannot modify the message and attach a MAC that succeeds. A receiver may be able to masquerade as a legitimate sender by modifying the packet and attaching a MAC that succeeds but the random number "k" will not match. There is no way for any adversary or malicious receiver to make a packet with a matching value of "k" as it does not possess the asymmetric encryption key, which is a secret known only to the sender(s).

MESSAGE INSERTION:

An adversary may insert messages in the stream. There is no way for an adversary to create messages which will be authenticated as legitimate packets. Inserted messages will fail authentication and will be dropped by the receivers.

MESSAGE DELETION:

An adversary may delete a message from the stream. Deletion attacks are not addressed by this scheme. It only deals with validation messages that are not deleted. However deleting some messages does not stop the authentication scheme from validating packets that have been delivered successfully.

EAVESDROPPING:

This chapter deals with data source authentication and not with confidentiality but the SIM-KM scheme uses proxy encryptions to secure data. It uses asymmetric keys and only receivers with legitimate decryption keys will be able to decipher the packet.

DENIAL OF SERVICE:

A malicious node can insert packets into the stream to slow down the receiver thus mounting a denial of service attack as the receiver will have to verify the packets. A node that is not part of the multicast session will not be able to make packets with matching MACs and thus will not be able to slow down the receiver as the MAC verification process does not involve decrypting the packet. A malicious receiver can slow down another receiver by inserting packets with matching MACs as it knows the symmetric key but the packet will still be detected as a bogus packet.

A malicious sender can send packets to receivers because it possesses the encryption key as well. In this case the denial of service attack cannot be mitigated and the only solution would be to have different encryption keys for different senders. However as discussed in Section 3 senders are trusted entities and are not likely to be disrupting the service themselves. Consider a scenario where multicast is used for conferencing

where there are multiple non-trusted senders and authentication is required for such a case then the only solution is to have a separate key pair for each sender. Our scheme would work well is such as case as well. The number of key pairs required will be small as there are few senders.

8.3 ANALYSIS

Let us consider a scenario where multicast is used to deliver stock updates to a large number of receivers. Some of these receivers may be PDAs, wireless handsets with limited resource and small bandwidths hence it rules out the possibility of using computationally intensive schemes or schemes with high communication overhead.

8.3.1 COMPARISON WITH OTHER SCHEMES

We compare our scheme with the Multiple MACs scheme, TESLA and Merkle hash schemes. We have not compared schemes that do not tolerate packet loss, are prone to message insertion attacks, etc., because these solutions are not suitable for use in the Internet.

Table 6 presents a comparison of these schemes. As seen the overhead caused by our proposal is either smaller or comparable when compared to different schemes. It does not suffer from collusion attacks or adversarial packet loss attacks, and does not require any time synchronization. It also does not require multicast data to be known in advance and works well with real time data. Given the basic traffic encryption scheme, this scheme adds only the overhead of a MAC computation. The proposed

scheme works better than existing schemes when other scenarios are considered such as Pay-per-view TV, online teaching, news feeds, etc.

Table 6: Overhead Comparison

Scheme	Overhead Per Packet	Need Synch.	Comparison
Multiple MACs Scheme	k bits where k depends on the size of the largest malicious receiver coalition	No	Suffers from collusion attack, needs to calculate k MACs before sending out every packet
TESLA	MAC + K_m where K_m is sent once every time period	Yes	Suffers from change in network propagation delay
Merkle Hash Scheme	n*(s + h log n) where n is the number of packets in the data stream, s is the signature size and h is the hash size	No	Suffers from Signature Flooding Attack, Adversarial Packet Loss Attacks, all messages need to be known in advance on sender side
Proposed Scheme	MAC + counter + 2*random number	No	Requires a single MAC computation, does not suffer from known attacks

The proposed scheme suffers from the drawback that it is only advantageous when data confidentiality is also required.

8.3.2 VALIDATION

We have used PROMELA (PROcess MEta LAnguage) [25] to specify the validation model and then used a tool, SPIN (Simple Promela INterpreter) [26] to validate our model. The model is designed so that it is simple but considers all the attacks listed in section 8.2. The model consists of one sender, one intermediate adversary and a receiver. The sender sends data packets. The intermediate adversary randomly modifies messages, inserts new messages, deletes messages and replays messages. Modified messages include messages with wrong MACs, wrong value of "k" in clear

text and altered data. Inserted messages include messages that were simply created and added to the data stream as well as messages with correct MACs. Replayed messages are messages that were saved from the data stream and were added to the stream after different time periods. The probability with which the intermediate adversary introduces errors can be controlled. The receiver verifies each packet received to establish if the packet has been sent by the sender or has been inserted/modified/replayed by the adversary.

XSPIN is used to specify the high level model written in PROMELA. As a preliminary check different random simulations were performed with different SPIN options and no errors were found. The verifier was compiled using the exhaustive search option. This option causes a state space search of all possible states and message timings of the modelled processes. Then, the verifier was executed and the output confirmed that our model is free from errors and there are no assertion violations, invalid end states or unreachable states in the design.

We compared the packets modified by the intermediate adversary to those detected by the receiver as packets having errors. In each case it was found that the packets that were randomly inserted, modified or replayed by the adversary were successfully detected by the receiver. The probability of errors introduced by the intermediate adversary was varied from 0 to 100% and it was found that the receiver was able to detect errors independent of the number of messages the adversary modified, inserted, replayed or deleted. The model was found to be free from errors, assertion violations, invalid end states or unreachable states in all cases.

9 MULTICAST POLICY MANAGEMENT REQUIREMENTS

Multicast security policies must represent, or contain, more information than a traditional peer-to-peer policy. In addition to representing the security mechanisms for the group communication, the policy must also represent the rules for the governance of the secure group. For example, policy would specify the authorization level necessary in order for an entity to join a group. More advanced operations would include the conditions when a group member must be forcibly removed from the group, and what to do if the group members need to resynchronize because of lost key management messages.

The translation of policy rules from one data model to another is much more difficult in a multicast group environment. This is especially true when group membership spans multiple administrative domains. Policies specified at a high level with a Policy Management tool must be translated into more precise rules that the available security policy mechanisms can both understand and implement. When dealing with multicast communication and its multiple participants, it is essential that the individual translation performed for each participant result in the use of a mechanism that is interoperable with the results of all of the other translations. Typically, the translation from high-level policy to specific policy objects must result in the same objects in order to achieve communication between all of the group members. The requirement that policy translation results in the same objects places constraints on the use and representations in the high-level policies.

Over the years various schemes have been developed for secure group communications. These schemes do not address the issue of policy representation but perform certain tasks that administer policy. None of these have found wide acceptance due to the lack of ability to handle wide ranges of users and environments. They do not express policies clearly and the policies cannot be modified explicitly. Policy tokens [44] cannot be viewed and modified easily by system administrators as there are no tools available to represent, interpret and display them visually. The goal of this paper is to present a policy representation scheme for secure group communications. It should have the ability to express policies clearly and should possess the possibility of modifying policies or creating new policies depending upon the context of the application.

As multicast groups may be distributed throughout the Internet, a system administrator should have the ability to express, monitor and enforce group security policies. There are some policy specification languages but they do not meet the requirements of a policy specification language for multicast communications [45]. They have not found wide acceptance and may be extended to suit policy specification for multicast but are unlikely to find acceptance due to cumbersome representation, need for specialized tools or modules to create, edit and parse these specifications, inability to adapt to rapid changes and other limitations. In this section we describe how XML can be used for policy representation in group communications.

This policy representation format addresses how policies can be represented in group communications so that key servers, policy servers, sender, receivers and other

entities can interact [46]. Our scheme fits within the MSEC framework and addresses the issue of policy representation for secure group communication. It is flexible, adaptable, does not require development of tools or modules to create, edit or parse policies. It can be easily understood by administrators and users and can easily be extended to include future changes and developments.

9.1 POLICY REPRESENTATION

The first element of any policy management scheme is the expression of policies. These policies must clearly establish the identity of the owner, the issuer, the period of validity, the policy assigned to an entity. All this must be cryptographically signed by the issuer so that forged certificates are not used as real certificates. This digital certificate will be known as a policy certificate as this certificate encodes the policy settings assigned to the user by the group manager.

X.509 [24] is the most widely used certificate standard today used in web browsers (Netscape Navigator and Microsoft Internet Explorer) that support the SSL (Secure Socket Layer), various code signing schemes (Java Archives and Microsoft Authenticode), secure e-mail standards (PEM and S/MIME), etc. Our policy representation closely follows the X.509 standard and has policies expressed in Extensible Markup Language (XML) [47]. XML is a rapidly maturing technology with powerful real-world applications, particularly for management, display and organization of data. XML describes a syntax that can be used to create a language. This enables the representation of certificates in a short and concise form and makes it

easy to parse and understand. The advantages of using XML for policy representation are as follows:

- The ability to edit and parse XML policy certificates across different platforms.
- The ability to present XML policy certificates in different formats that provide an intuitive and useful representation for administrators and users. Extensible Stylesheet Language Transformations (XSLT) stylesheets can be associated with the certificates to transform XML documents. Thus instead of having illegible certificates, which can only be understood by machines, it would be possible to represent policy certificates in graphic formats that can be easily understood by humans. Also it is possible to create stylesheets that can transform the XML policy file to other policy file formats.
- The existence of XML editors, parsers and translators that are freely and widely available. They are already being used for communication purposes. This removes the requirement for developing tools for such purposes.
- The existence of intelligent agents that can parse policy certificates. This can be used for reporting conflicts and for negotiating security settings.

9.2 POLICY VALIDATION

A policy certificate being a digital certificate is cryptographically signed and this signature can be verified to ensure that the certificate is not forged. However each entity in a multicast environment has a local requirement on group sessions. These policies must be matched with the group policies specified in the certificate to ensure that the policy settings will be compliant with the entity's own requirements before

they are implemented. This will ensure that there are no undesirable effects. Document Type Definitions (DTDs) can be used for validating the XML policy file. This will enable errors to be identified easily. The purpose of a DTD is to define the legal building blocks. This can be done inline in the XML document or as an external reference. With a DTD, independent groups of users can agree to use a common DTD. An entity can use its own standard DTD to verify if the policies requested to be enforced by other entities are valid.

9.3 POLICY REALM

In group communications, a security policy defines the security features to be implemented by the group entities. There can be two types of policies:

- ***Control Policies***: They outline the policies required for the functioning of the group according to the requirements of the group and the group manager. Examples of control policies are member authentication, member join, member leave, billing and role delegation.
- ***Data Policies***: They outline the policies required to protect the multicast data such as data security, integrity and authentication policies and re-keying policies.

Some of the control policies are as follows:

MEMBER AUTHENTICATION POLICY:

This policy specifies how to authenticate members who want to join the group. For corporate meetings it is necessary to have strict authentication and to establish the true

identity of the member. For teaching groups and pay-per-view TV, it is enough to have a subscription member and there is no need to establish the true identity of the member. For internet radio it is not necessary to perform any authentication and any loose subscription will suffice.

MEMBER JOIN POLICY:

This policy specifies what should happen when a member is authenticated and is allowed the join the group. How the keys should be given to the member, is there a need for changing keys to ensure perfect forward secrecy, what keys should the remaining group members use now, if keys need to be changed then can joins be bunched and if joins can be bunched then how many joins or for how long should the entity wait. The decision will be made depending upon the group join policy specified in the policy certificate created by the group manager and specified by the system administrator depending upon group characteristics and requirements.

MEMBER LEAVE POLICY:

It is similar to the group member join policy and specifies what should happen when a group member leaves the multicast group or is forcibly removed from the group.

ROLE DELEGATION POLICY:

This policy specifies what actions a member may perform after being admitted to the group. Depending upon the group requirements a member may or may not be allowed to send data to the group, to distribute keying material, to modify policies, to admit or ban members and to monitor group statistics.

BILLING POLICY:

In certain classes of applications such as Internet TV, it is necessary to accurately identify users who are viewing multicast content so that they can be charged for the service. It is necessary to keep track of who is listening to the multicast content, for how long and at what times. This will also help to identify the popularity of the contents and in turn help generate advertisement revenue for the content providers. In some other classes of applications such as pay-per-view TV, it is necessary to keep track of who is listening but it is not needed to keep track for how long as the user will be charged for the entire content he subscribes to watch even if the user logs out. In some other applications such as Internet radio it may not be necessary to keep track of who is listening or for how long.

Some of the data policies are as follows:

DATA SECURITY POLICY:

This policy is necessary to specify how data will be protected. It identifies the encryption scheme or the type of confidentiality setting to protect the data from unauthorized access.

DATA AUTHENTICATION POLICY:

This policy is required to ensure that the data received have originated from a valid source. Two flavours of data authentication may be used either to ensure that the message was sent by a member of the group or to ensure that the message was sent by particular member.

DATA INTEGRITY POLICY:

This policy is required to ensure that no party modifies the data while they are in transit. Schemes to detect modification and achieve message integrity may be used.

RE-KEYING POLICY:

This policy specifies how frequently a re-key must occur. They may occur once the lifetime of a key has expired or they may be induced by membership changes.

9.4 POLICY ARCHITECTURE

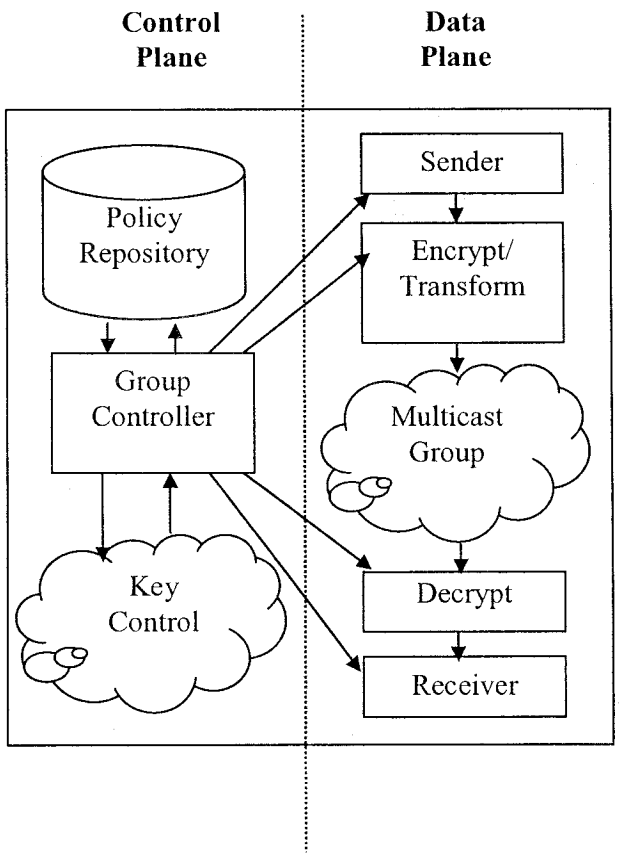

Figure 23: Policy in Secure Group Communications

Group policies are created by the group owner and stored in the policy repository. This may be done by the system administrator initiating the group creation or by the node depending on session requirements. Figure 23 represents how the policy architecture fits in with the SIM-KM architecture. The group controllers interact with the policy repository. They obtain the certificates necessary for the group from the policy repository and verify them. Once verified, these policies are validated against local group policies as in section 9.2. These policies are then implemented by the group controllers. The group controllers distribute these policies to the group members in a similar fashion. They thus act as policy repositories for the group members ensuring that the policy management infrastructure is distributed throughout the network. Any change in group policy must be communicated to the group controllers. The group controllers may also verify the policies from the node where they obtained the group policy certificates after a certain time interval to ensure that the policies are still valid.

9.5 XML REPRESENTATION

The XML representation defines a group and the rules to be followed by the members of that group. Group is used to identify the multicast group for which the XML certificate is valid. Rules define the various settings for group members and the functions they are allowed to perform. A sample XML document is shown in figure 24. The structure of the part representing multicast policy in the XML document is as follows:

```
<GROUP>
    <IDENTIFIER>
        <!---->
        <!--Group Name-->
```

```
        <!---->
    </IDENTIFIER>
    <REGISTRATION>
        <!---->
        <!--Rules-->
        <!---->
    </REGISTRATION>

    <RE-KEY>
        <!---->
        <!--Rules-->
        <!---->
    </RE-KEY>

    <DATAMANAGER>
        <!---->
        <!--Rules-->
        <!---->
    </DATAMANAGER>
</GROUP>
```

The group for which the policy is valid is identified by a group name in the identifier tag. Rules can be of three different types: rules for registration, rules for re-key and rules for data management. Rules are of the structure:

```
<RULENAME>

.....

</RULENAME>
```

RULENAME is used to identify the rule. If two rules appear with the same RULENAME then the one appearing first should be applied if it is supported by local policy.

Rules for registration provide a list of acceptable registration and deregistration policies and mechanisms to allow group members to join and leave the group. If the registration rule is blank then the group does not support the feature and anybody can join or leave the group and the group works as an "anyone can send anyone can

receive" group. If multiple mechanisms are available, then they are listed in terms of priority of usage by the group owner.

A group member may use any of the listed mechanisms preferably one that is higher in the list and is supported by local policy. When a rule is listed the group member is forced to use one of the rules. Re-key rules list the re-key mechanism and may also list frequency of re-key messages. Data management rules provide the mechanisms how data are handled such as encryption and authentication schemes. If multiple rules are provided for any scheme then the order of the list implies the order of encapsulation of data.

9.5.1 REGISTRATION

The protocol for registration and deregistration is defined in the MSEC architecture. It is the process by which a prospective group member proves its identity to a group controller and obtains the required keys and information required to join the group. The deregistration process allows a group member to inform a group controller that it no longer wants to be part of a multicast group. This can also happen due to a join state not being refreshed or a group member being removed from the group.

9.5.2 RE-KEY

Re-key rules state the re-key mechanism being used, the re-key interval and other re-key specific information. It also states when re-key or key changes may be initiated. For example a re-key maybe needed to ensure perfect forward or backward secrecy

when a new group member joins or leaves a group. Samples rules for re-key are as follows:

```
<RE-KEYPROTOCOL>
        SIM-KM
</RE-KEYPROTOCOL>
<FORWARDSECRECY>
        YES
</FORWARDSECRECY>
<BACKWARDSECRECY>
        YES
</BACKWARDSECRECY>
<RE-KEYINTERVAL>
        60
</RE-KEYINTERVAL>
<EVENTINTERVAL>
        10
</EVENTINTERVAL>
```

The example shows that SIM-KM will be the protocol for re-keying. Key changes must take place to ensure perfect forward and backward secrecy. The group controllers and members must see a re-key message every 60 seconds. Key changes may be delayed to avoid having continuous key changes up to a maximum of 10 events. Thus a maximum of 10 joins and leaves may be bunched together to initiate a key change.

9.5.3 DATA MANAGEMENT

The data sent is transformed to provide security. A cryptographic algorithm is applied over a set of data and keys to generate a cipher text. This renders the data useless for malicious users. A hash is appended to the data using a collusion-resistant and non-invertible hash function. This ensures authenticity and message integrity. There are a number of schemes to provide multicast confidentiality, authenticity and message integrity. SIM-KM uses proxy encryptions in a novel method to ensure confidentiality. It provides source authentication, which most schemes are unable to

provide or which involves enormous overhead to provide. Sample rules for data management are as follows:

```
<ENCR>
        AES
</ENCR>
<AUTH>
        SHA1
</AUTH>
<ENCRKEYSIZE>
        64
</ENCRKEYSIZE>
<AUTHKEYSIZE>
        32
</AUTHKEYSIZE>
```

The numbers and mechanisms mentioned are samples and actual values will depend on the nature and dynamics of the multicast group. ENCR lists the encryption mechanism used to protect data. AUTH lists the method used to authenticate the data. ENCRKEYSIZE gives the size of the encryption key in bytes and AUTHKEYSIZE gives the size of the authentication key in bytes. Other fields may be added depending upon the context of usage. Fields may be left blank. For example if authentication is not required then the authentication field may be left blank.

The complete format of the XML file is shown in figure 24. It closely follows the X.509 notation. Some of the components are mandatory such as the issuer's public key, the subject's public key, the certificate type and version of the certificate. It also includes a serial number to indicate revocation if it is included in the issuer's revocation list. An expiry date to indicate the validity of the certificate is also necessary.

```
<?xml version="1.0"?>
<!DOCTYPE root..Certificate SYSTEM "C:\Thesis\policy\cpm_DTD.dtd">
<root..Certificate>
    <Certificate.tbsCertificate>
        <TBSCertificate.version>2</TBSCertificate.version>
```

```
<TBSCertificate.serialNumber>1234</TBSCertificate.serialNumber>
<TBSCertificate.signature>

<AlgorithmIdentifier.algorithm>1.2.840.113549.1.1.5</AlgorithmIdentifier.algorithm>
        <AlgorithmIdentifier.parameters>
                <_NULL/>
        </AlgorithmIdentifier.parameters>
</TBSCertificate.signature>
<TBSCertificate.issuer>
        <Name.rdnSequence>
                <RDNSequence._>
                        <RelativeDistinguishedName._>

<AttributeTypeAndValue.type>2.5.4.10</AttributeTypeAndValue.type>
                                <AttributeTypeAndValue.value>

<_PrintableString>University1</_PrintableString>
                                </AttributeTypeAndValue.value>
                        </RelativeDistinguishedName._>
                </RDNSequence._>
                <RDNSequence._>
                        <RelativeDistinguishedName._>

<AttributeTypeAndValue.type>2.5.4.11</AttributeTypeAndValue.type>
                                <AttributeTypeAndValue.value>
                                        <_PrintableString>Division1</_PrintableString>
                                </AttributeTypeAndValue.value>
                        </RelativeDistinguishedName._>
                </RDNSequence._>
        </Name.rdnSequence>
</TBSCertificate.issuer>
<TBSCertificate.validity>
        <Validity.notBefore>
                <Time.utcTime>040901000000Z</Time.utcTime>
        </Validity.notBefore>
        <Validity.notAfter>
                <Time.utcTime>040910000000Z</Time.utcTime>
        </Validity.notAfter>
</TBSCertificate.validity>
<TBSCertificate.subject>
        <Name.rdnSequence/>
</TBSCertificate.subject>
<TBSCertificate.subjectPublicKeyInfo>
        <SubjectPublicKeyInfo.algorithm>

<AlgorithmIdentifier.algorithm>1.2.840.113549.1.1.1</AlgorithmIdentifier.algorithm>
                <AlgorithmIdentifier.parameters>
                        <_NULL/>
                </AlgorithmIdentifier.parameters>
        </SubjectPublicKeyInfo.algorithm>
        <SubjectPublicKeyInfo.subjectPublicKey encoding="base64"
        length="2160">MIIBCgKCAQEA0bt/sF14bMMET5Y06mD5iLyKL/pAyIZHGfERJHWLPtB
        5CDc2N55/Hp4d3puVgL0PtKqiyOagPY5yn9VW5df8WEYoTFPwTM/er4C5bPAhV2s8AW
        pU+7eWqrXojaN1Iu2GHnMmCVUxk6COqupopCRkzLkEUhJH060aLORrJ2aBOaA/qsRW
        RR0jhNJ0uVwQWrd5NsOcA84StjXPcSNvZ3muDZJQee5cFXG7TeFOBl2CRuWpyjQBQ9
        6I43LrRw8oAm1DYO8GCKI6rOlF0aSZmFXbJmUirxB6vIAAMx67oJ1rcrFLRAJhJkDBysA
        hgdcvNYIl9c1sQuESu9rDRyPNZhqIxwIDAQAB</SubjectPublicKeyInfo.subjectPublicK
        ey>
</TBSCertificate.subjectPublicKeyInfo>
<TBSCertificate.extensions>
        <Extensions._>
                <Extension.extnId>2.5.29.35</Extension.extnId>
                <Extension.critical>TRUE</Extension.critical>
                <Extension.extnValue>
                        <root..AuthorityKeyIdentifier>
                                <AuthorityKeyIdentifier.keyIdentifier encoding="base64"
length="160">BXmVx2FIl9b2QXhkiwDZaEb9K2j=</AuthorityKeyIdentifier.keyIdentifier>
                        </root..AuthorityKeyIdentifier>
                </Extension.extnValue>
        </Extensions._>
        <Extensions._>
```

```
                        <Extension.extnId>2.5.29.14</Extension.extnId>
                        <Extension.critical>FALSE</Extension.critical>
                        <Extension.extnValue>
                                <root..AuthorityKeyIdentifier>
                                        <AuthorityKeyIdentifier.keyIdentifier encoding="base64"
length="160">BXmVx2FII9b2QXhkiwDZaEb9K2j=</AuthorityKeyIdentifier.keyIdentifier>
                                </root..AuthorityKeyIdentifier>
                        </Extension.extnValue>
                </Extensions._>
        </TBSCertificate.extensions>
    </Certificate.tbsCertificate>
    <Certificate.signatureAlgorithm>
        <AlgorithmIdentifier.algorithm>1.2.840.113549.1.1.5</AlgorithmIdentifier.algorithm>
        <AlgorithmIdentifier.parameters>
                <_NULL/>
        </AlgorithmIdentifier.parameters>
    </Certificate.signatureAlgorithm>
    <Certificate.signature encoding="base64"
    length="2048">I8NyN3GOh/V7VuJs3ILBLaaH6v+KMCbXY9kNs3/a0aeRXKrFSKOe9evUn3OCjNGyv
    CKmDrLhmzSUGKKp0x6SuzCZkgB16xhCoZSu2awBjFAcC/n3SjjHKvGGxofH/c7+I2Uss+tK7bmxPm
    mq1I1ahMgfH2sHuwmrNCw4QXNExv954PlIdad5XQKWDkMHdPt24GI0GmRhptHdKkgKts6kbs5CZIz
    35BMCCB3ZO2tRnCn5RN2IJ1nJdNy4M7yOWMZJbaiY3S2hRkjnrr0dn+b9YnG1J83ydeOFo2Bbxkrhg
    mopiR/mO6hmT5/Gr5WFvDdubR4clYQHgYkz61jrR3POQw==</Certificate.signature>
    <Group>
        <Identifier>pay_per_view</Identifier>
        <Registration>password</Registeration>
        <Re-key>
                <RE-KEYPROTOCOL>SIM-KM</RE-KEYPROTOCOL>
                <FORWARDSECRECY>YES</FORWARDSECRECY>
                <BACKWARDSECRECY>YES</BACKWARDSECRECY>
                <RE-KEYINTERVAL>60</RE-KEYINTERVAL>
                <EVENTINTERVAL>10</EVENTINTERVAL>
        </Re-key>
        <DataManager>
                <ENCR>AES</ENCR>
                <AUTH>SHA1</AUTH>
                <ENCRKEYSIZE>64</ENCRKEYSIZE>
                <AUTHKEYSIZE>32</AUTHKEYSIZE>
        </DataManager>
    </Group>
</root..Certificate>
```

Figure 24: XML Policy File

A DTD was created with the XML document. To demonstrate the flexibility of creating, editing and interpreting the policy files in XML format, a generic XML editor was used to view and edit the XML policy file. We used XML Spy (Copyright © 1998 – 2004 Altova GmbH) for viewing and editing the XML file. The file was verified to be well-formed and validated with the DTD.

10 CONCLUSIONS AND FUTURE WORK

Multicast is a promising technology to provide large-scale efficient delivery of data to a large number of users. IP multicast uses the least network bandwidth when compared to any technology to deliver source data to multiple receivers. In this thesis it has been shown that in order to ensure large scale deployment and use of multicast, new security infrastructures are required. Traditional unicast security solutions are unable to meet the requirements for multicast security. The large number of users, dynamic nature of multicast groups and best effort delivery service make multicast security difficult to achieve. MSEC had divided the tasks associated with multicast security into three major areas: Key Management for Data Confidentiality, Data Authentication and Policy Management. This thesis presents solutions for each of these areas.

A secure key management scheme for multicast communications has been presented. SIM-KM is simple, scalable and works with the existing proposed group key management architecture [10] proposed by the IETF. It is a seamless extension and fits with the group security architecture. The solution does not place complete trust in intermediate components. Moreover the intermediate components are unable to see the clear text messages while they modify the cipher text. The method ensures that the number of messages are *O(logN)* or less, where N is the size of the group. The solution is also dynamic and flexible giving it the possibility of having group controllers depending on the sensitivity of the multicast data. For highly classified information the group manager may prefer to have no group controllers at all and will perform all operations itself. For other types of multicast the group manager may

choose to share access lists and keys with other group controllers giving it the flexibility of have a widely distributed design and thus very high scalability. Also the group manager may have trusted components in some parts of the network and may designate only those group controllers with the ability to authenticate participants and distribute keys. The protocol uses the concept of aggregation to avoid repeated changes to the key. This feature is dynamic and the time interval for aggregation can be changed dynamically during the lifecycle of the multicast group. The scheme was modelled using PROMELA and validated using SPIN. A performance evaluation was performed to show that the scheme is superior to other similar schemes.

An efficient robust scheme for multicast group authentication was presented. It can be used with any asymmetric multicast data security protocol for multicast group security (such as SIM-KM) to perform multicast data source authentication. The authentication scheme has no delays, requires no time synchronization, is collusion-resistant, and does not have a large packet overhead. It is simple, works independent of packet losses and is not resource intensive. It is a robust, efficient and scalable solution for multicast data source authentication. The scheme is flexible and does not place any restriction on which asymmetric encryption scheme to use or on the size of the keys and the choice of MACs. The scheme was modelled using PROMELA and validated using SPIN, which showed the scheme is successful in the face of all possible known message authentication attacks.

A simple straightforward method to represent multicast policy was presented. XML provides a flexible method for definition and implementation of a wide range of secure group policies. It has been shown how XML can be used to define multicast

group policies and how XML documents modelled on the same structure as X.509 certificates can be created, edited and verified using available XML editors. Some critical group policies were also identified.

10.1 FUTURE DIRECTIONS

Future work includes the development of an implementation of SIM-KM along with the data authentication and message integrity scheme for testing the delays and actual performance parameters involved with modifying the cipher texts at different group controllers.

A billing protocol or mechanism is needed for Internet Service Providers to keep track or record usage of multicast services to generate appropriate revenues. It is necessary to identify the different group policy settings possible for different applications and probably have a tool for network administrators to choose multicast group settings and automatically generate group certificates in XML. To test how XML documents are interpreted by a policy engine and how to verify if policy settings are being implemented by a local controller would be the next step.

BIBLIOGRAPHY

[1] B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan, Internet Group Management Protocol, Version 3, *RFC3376*, October 2002.

[2] R. Vida, L. Costa, Multicast Listener Discovery Version 2 (MLDv2) for IPv6, *RFC3810*, June 2004.

[3] T. Pusateri, Distance Vector Multicast Routing Protocol, Internet Draft, <draft-ietf-idmr-dvmrp-v3-11.txt>, *Work in Progress*, October 2003.

[4] J. Moy, Multicast Extensions to OSPF, *RFC1584*, March 1994.

[5] B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised), Internet Draft, <draft-ietf-pim-sm-v2-new-11.txt>, *Work in Progress*, October 2004.

[6] S. Mittra, Iolus: A Framework for Scalable Secure Multicasting, *ACM SIGCOMM*, Cannes, France, September 1997.

[7] P. McDaniel, H. Harney, A. Colgrove, A. Prakash, P. Dinsmore, Multicast Security Policy Requirements and Building Blocks, Internet Draft, <draft-irtf-smug-polreq-00.txt>, *Work in Progress*, May 2001.

[8] P. McDaniel, H. Harney, P. Dinsmore, A. Prakash, Multicast Security Policy, Internet Draft, <draft-irtf-mcast-policy-01.txt>, *Work in Progress*, November 2000.

[9] T. Hardjono, B. Weis, The Multicast Group Security Architecture, *RFC3740*, March 2004.

[10] M. Baugher, R. Canetti, L. Dondeti, F. Lindholm, Multicast Security (MSEC) Group Key Management Architecture, *RFC4046*, April 2005.

[11] M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman, The Secure Real-time Transport Protocol (SRTP), *RFC 3711*, March 2004.

[12] H. Harney, C. Muckenhirn, Group Key Management Protocol (GKMP) Architecture, *RFC2094*, July 1997.

[13] H. Harney, C. Muckenhirn, Group Key Management Protocol (GKMP) Specification, *RFC2093*, July 1997.

[14] A. Ballardie, Scalable Multicast Key Distribution (SKMD), *RFC1949*, May 1996.

[15] C. K. Wong, M. Gouda, S. S. Lam, Secure Group Communications Using Key Graphs, *ACM SIGCOMM* , Vancouver, B.C., September 1998.

[16] J. Huang, S. Mishra, Mykil: A Highly Scalable and Efficient Key Distribution Protocol for Large Group Multicast, *IEEE GLOBECOM*, San Francisco, CA, December 2003.

[17] A. T. Sherman, D. A. McGrew, Key establishment in Large Dynamic Groups Using One-Way Function Trees, *IEEE Transactions on Software Engineering, Volume 29, Issue 5*, May 2003.

[18] Y. Challal, H. Bettahar, A. Bouabdallah, SAKM: A Scalable and Adaptive Key Management Approach for Multicast Communications, *ACM SIGCOMM Computer Communications Review, Volume 34, Issue 2*, April 2004.

[19] S. Setia, S. Koussih, S. Jajodia, E. Harder, Kronos: A Scalable Group Re-keying Approach for Secure Multicast, *IEEE Symposium on Security and Privacy*, May 2000.

[20] Ritesh Mukherjee, J. William Atwood, SIM-KM: Scalable Infrastructure for Multicast Key Management, *IEEE Conference on Local Computer Networks*, Tampa, FL, November 2004.

[21] Ritesh Mukherjee, J. William Atwood, Proxy Encryptions for Secure Multicast Key Management, *IEEE Conference on Local Computer Networks*, Bonn/Koenigswinter, Germany, October 2003.

[22] M. Blaze, G. Bleumer, M. Strauss, Divertible Protocols and Atomic Proxy Cryptography, *Eurocrypt*, Helsinki, Finland, 1998.

[23] Y. Dodis, A. Ivan, Proxy Encryption Revisited, *Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2003.

[24] ITU-T Recommendation X.509 (2000) | ISO/IEC 9594-8:2001, Information Technology – Open Systems Interconnections – The Directory: Public-key and attribute certificate frameworks.

[25] G. J. Holzmann, Design and Validation of Computer Protocols, *Prentice Hall*, 1991.

[26] G. J. Holzmann, The Model Checker SPIN, *IEEE Transactions on Software Engineering, Volume 23, Issue 5*, 1997.

[27] B. Kaliski, J. Staddon, PKCS #1: RSA Cryptography Specifications Version 2.0, *RFC2437*, 1998.

[28] R. D. Silverman, A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths, *Bulletin 13, RSA Laboratories*, 2001.

[29] A. K. Lenstra, E. R. Verheul, Selecting Cryptographic Key Sizes, *Journal of Cryptology*, Volume 14, 2001.

[30] E. W. Zegura, K. Calvert, S. Bhattacharjee, How to model an Internetwork, *IEEE INFOCOM*, San Francisco, CA, 1996.

[31] K. Fall, K. Varadhan, The ns manual, *http://www.isi.edu/nsnam/ns/doc/*, August 2000.

[32] Ritesh Mukherjee, J. William Atwood, A Comparative Analysis of SIM-KM for Group Key Management, *IASTED International Conference on Communication and Computer Networks*, Cambridge, MA, U.S.A., November 2004.

[33] R. C. Chalmers, K. C. Almeroth, On the Topology of Multicast Trees, *IEEE/ACM Transactions on Networking, Volume 11(1)*, 2003.

[34] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, A Taxonomy and some Efficient Constructions, *IEEE INFOCOM*, Kobe, Japan, March 1997.

[35] A. Perrig, J. D. Tygar, D. Song, R. Canetti, Efficient Authentication and Signing of Multicast Streams over Lossy Channels, *IEEE Symposium on Security and Privacy*, Berkeley, CA, May 2000.

[36] R. Gennaro, P. Rohatgi, How to Sign Digital Streams, *CRYPTO*, Santa Barbara, CA, August 1997.

[37] A. Perrig (November 2001), The BiBa One-Time Signature and Broadcast Authentication Protocol, *ACM Conference on Computer and Communications Security*, Philadelphia, Pennsylvania, November 2001.

[38] C. K. Wong, S. S. Lam, Digital Signatures for Flows and Multicasts, *International Conference on Network Protocols (ICNP)*, Austin, Texas, October 1998.

[39] A. Pannetrat, R. Molva, Efficient Multicast Packet Authentication, *Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2003.

[40] P. Golle, N. Modadugu, Authenticating Streamed Data in the Presence of Random Packet Loss, *Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2001.

[41] B. Weis, The Use of RSA Signatures with ESP and AH, Internet Draft, <draft-ietf-msec-ipsec-signatures-04.txt>, *Work in Progress*, February 2004.

[42] Ritesh Mukherjee, J. William Atwood, Multicast Group Authentication, *Accepted for publication in the Network Control and Engineering for QoS, Security and Mobility (NETCON)*, Lannion, France, November 2005.

[43] E. Rescorla, B. Korver, Guidelines for Writing RFC Text on Security Considerations, *RFC3552*, July 2003.

[44] A. Colegrove, H. Harney, Group Policy Token v1: Group Policy Token V1 with Application to GSAKMP, <draft-ietf-msec-policy-token-sec-03.txt>, *Work in Progress*, July 2005.

[45] A. Meissner, S. B. Musunoori, L. Wolf, MGMS/GML – Towards a New Policy Specification Framework for Multicast Group Integrity, *IEEE International Symposium on Applications and the Internet*, Tokyo, Japan, January 2004.

[46] Ritesh Mukherjee, J. William Atwood, XML Policy Representation for Secure Multicast, *IEEE SouthEastCon Conference*, Fort Lauderdale, FL, April 2005.

[47] Extensible Markup Language (XML) 1.0 (Third Edition), *http://www.w3.org/TR/2004/REC-xml-20040204/*, February 2004.