ABSTRACT

Title of Dissertation: MULTI-USER SECURITY FOR MULTICAST COMMUNICATIONS

Yan Sun, Doctor of Philosophy, 2004

Dissertation directed by: Professor K. J. Ray Liu
Department of Electrical and Computer Engineering

The ubiquity of communication networks is facilitating the development of wireless and Internet applications aimed at allowing users to communicate and collaborate amongst themselves. In the future, group-oriented services will be one of the dominant services that facilitate real-time information exchange among a large number of diverse users. However, before these group-oriented services can be successful deployed, technologies must be developed to guarantee the security of the information and data exchanged in group communications.

Among all security requirements of group communication, access control is paramount as it is the first line of defense that prevents unauthorized access to the group communication and protects the value of application data. Access control

is usually achieved by encrypting the data using a key that is shared among all legitimated group members. The problem of access control becomes more difficult when the content is distributed to a dynamic group with user joining and leaving the service for a variety of reasons. Thus, Group Key Management is required to achieve key update with dynamic group membership.

Existing group key management schemes seek to minimize either the amount of rounds needed in establishing the group key, or the size of the key updating messages. They do not, however, considering the varying requirements of the users, the underlying networks or the applications. Those generic solutions of access control often yield large consumption of communication, computation and storage resources, especially for large groups with highly dynamic membership in heterogeneous networks. In addition, the design of existing key management schemes focus on protecting the application data, but introduces vulnerabilities in protecting the statistics of group membership information. This poses severe security concern in various group applications.

The focus of this dissertation is to design network-specific and application specific group key management and solve the security vulnerability of key management that reveals dynamic group membership information. This dissertation will present scalable group key management in heterogeneous wireless network, the hierarchical access control for multimedia applications, and a framework of securing dynamic group membership information over multicast. The main contribution of this dissertation is to advance the group key management research to achieve higher level of scalability and security.

MULTI-USER SECURITY FOR MULTICAST COMMUNICATIONS

by

Yan Sun

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2004

DEDICATION

To My Father, JingLun Sun

ACKNOWLEDGEMENTS

First of all, I would like to express deep appreciation to my advisor, Prof. K. J. Ray Liu. Five years ago, his class lectures stimulated my curiosity in research, which later led to one of the most important decisions in my life - pursuing the PhD degree. He led me onto the exciting journey of academia. In every stage of my PhD study, his knowledge, vision, encouragement, and guidance pose significant influence on me and provide me with lifetime benefits. This dissertation could not have been a reality without his help and support.

I am grateful to the members of the DSP group. I would like to explicitly mention Dr. Wade Trappe, Dr. Zoltan Safar, Dr. Xiaowen Wang, and Dr. Jie Song, who helped me to start the research and gave me advices on all aspects of graduate study. In fact, one of the research ideas in this dissertation originated from the brainstorming with Wade. I am also indebted to the collaboration with Charles Pandana, Yinian Mao and Wei Yu. I want to thank Dr. Jane Wang, Hong Zhao, Dr. Weifeng Su, Dr. Zhu Han, Ji Zhu, and Johannes Thorsteinsson for sharing ideas and discussing with me. It is a true pleasure to work in the DSP group. I am proud to be a part of this diversified while strongly bound team.

I would like to thank Prof. Min Wu for sharing her experiences as a young faculty member, and for giving me valuable suggestions on research, teaching and academic career. In addition, I appreciate her support to working with her PhD students and undergraduate students in her class. I also want to thank Prof.

Anthony Ephremides, Prof. Sennur Ulukus, and Prof. Lawrence C. Washington for their valuable comments and suggestions on the thesis draft.

Finally, I want to take this opportunity to express my sincere appreciation to my parents and my husband, for their love and unconditional support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

Point-to-point communication has been the dominant form of computer network communication since the beginning of networking. However, with the explosive advancement of networking and information technologies that bring a large amount of users to communicate and collaborate, point-to-point communications faces severe scalability challenges in a variety of emerging applications [1].

- Digital video and audio multicast over Internet, such as movie-on-demand and video conferences.
- Widespread software distribution, such as anti-virus scanner update and security patch delivery.
- Disseminating real-time financial market information to a large audience with various devices, including PDAs, cell phones, computers etc.
- Transportation control where road traffic pattern or air traffic control information is distributed to many stations.

- Multi-player games involving thousands of users simultaneously interacting in a virtual game world.

Distributing information to a large audience cannot be achieved over point-to-point communications without causing congestion and wasting network resources. For group-oriented applications, *Multicast* is an essential mechanism to achieve scalable information distribution.

Multicast describes communication where information is sent from one or more parties to a set of other parties. In this case, information is distributed from one or more senders to a set of receivers, but not to all users. Multicast is different from broadcast where information is distributed to all users. The subtle difference between broadcast and multicast can be neglected in many context.

Significant advancement has been seen recently, in both the underlying multicast networking technology [2] as well as the deployment of applications utilizing multicast [1]. Already there are multicast services that stream stock quotes, and provide video and audio on demand. In the future, multicast applications will be running in wireless mobile environment, as consumers desire to have a similar suite of multimedia-intensive applications on their portable devices as they currently have available to them at their desktops.

Multicast has the advantage of efficient distributing information to thousands or even millions of users. However, multicast also creates opportunities for malicious packets to reach thousands or millions of users, and introduces difficulties in maintaining security with dynamic group membership. In this chapter, we will first review security issues in group communications, then discuss some drawbacks of existing security mechanisms, and finally summarize the contribution of this thesis.

1.2 Security Issues in Group Communications

In general, group communications consider the following security requirements.

Confidentiality non-group members cannot read the data

Integrity data cannot be modified or deleted in any unauthorized way

Authentication claimed sender is the actual sender

Access Control only authorized parties can access the group communications

Non-repudiation Recipient can prove what messages it receives

No denial-of-service No interference by un-authorized parties

1.2.1 Access Control and Data Confidentiality

Among all those requirements, *access control* is the first line of defense needed to protect the value of application data. A service provider may control access to content by encrypting the content using a key that is shared by all valid group members. The problem of access control becomes more difficult when the content is distributed to a group of users. Since group membership will most likely be dynamic with users joining and leaving the service for a variety of reasons, it is necessary to update the group key. The issues of key generation and update are addressed by *Key Management* protocols [3, 4]. In addition, encryption and key management together ensure data confidentiality because unauthorized entities do not possess the group key and cannot decrypt group communication.

The main problem of group key management is to update keys in dynamically changing group such that a receiver can decrypt the data only when he is a valid group member. In particular, a group key management protocol should satisfy the following security requirements.

- Group key secrecy - non-group members cannot obtain any group key.
- Backward secrecy - the join user cannot decrypt the content that was sent before his join.
- Forward secrecy - the departure/revoked user cannot decrypt the content that is sent after his deletion from the group.

To achieve forward and backward secrecy, the group key is updated after each member join and departure event, and the new key information is distributed to the legitimated group members . It is important to update and distribute keys in a secure, scalable and reliable way. In Chapter 1.4, these issues will be discussed in detail. Here we list the properties that a good key management scheme should have.

- Low communication, computation and storage overhead
- Scalability for large dynamic groups
- Reliable distribution of key update messages
- The ability of detecting dishonest group members and recovering from key generation failure.

1.2.2 Service Authentication and Verification

Multicast Authentication is another essential security mechanism that ensures data integrity and validates the source of the data. Authentication can be achieved in an asymmetric manner such as digital signatures. Although digital signatures can solve the authentication problem, they are inefficient due to their prohibitive computational overhead [5–7]. In point-to-point communications, data authentication is usually achieved through a symmetric manner where the sender and the receiver share a secret key to compute a message authentication code (MAC) of all communicated data. When a message with a correct MAC arrives, the receiver is assured that the sender generated that message. However, symmetric authentication is generally not secure in multicast because every group member knows the MAC key and can impersonate the sender. It is required that every receiver can verify the authenticity of messages without being able to generate authentic messages, which is asymmetric in nature. Currently, the most popular multicast authentication schemes use the principle of delayed key disclosure in order to achieve the asymmetry needed for multicast authentication using symmetric cryptography [8–12]. Those schemes require time synchronization between the source and the receivers. The main idea is to have the sender attach a MAC to each packet computed using a key known only to itself. The receivers buffer the received packets without being able to authenticate them. A short time after the delivery of the packets, the sender discloses the key and the receivers are later able to authenticate the packets. After the key is disclosed, the receivers will not accept the packages with MAC generated by this key. A representative of such authentication schemes is TESLA. Interested readers can refer to [11] for the details of TESLA and other multicast authentication schemes. In addition, non-repudiation problem and prevention of Denial-of-service attack in secure multicast communications are relatively new topics. Some studies can be found in [13–15]. In this dissertation, the focus is access control for secure group communications.

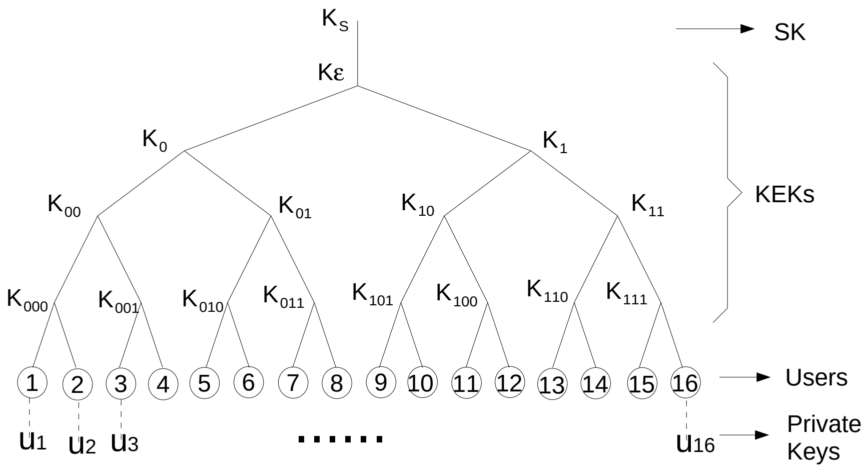


Figure 1.1: A typical key management tree

1.3 Key Management for Group Access Control

Key management schemes can be classified as centralized schemes and contributory schemes [16]. In centralized schemes, group members trust a centralized server, referred to as the key distribution center (KDC), which generates and distributes encryption keys [4, 16–24]. In contributory schemes, group members are trusted equally and all participate in the formation of the group key [25–33]. In this section, we introduce popular centralized and contributory key management schemes.

1.3.1 Centralized Key Management

The most common class of centralized key management schemes employ a tree hierarchy to maintain the keying material [3, 16–19]. As illustrated in Figure 1.1, each node of the key tree is associated with a key. The root of the key tree is associated with the session key (SK), K_s, which is used to encrypt the multicast content. Each leaf node is associated with a user's private key, u_i, which is only known by this user and the KDC. The intermediate nodes are associated with

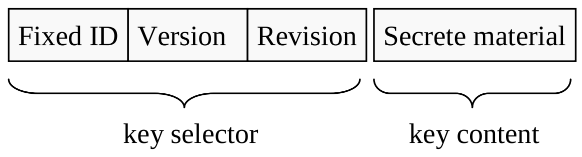


Figure 1.2: Key structure

key-encrypted-keys (KEK), which are auxiliary keys and only for the purpose of protecting the session key and other KEKs. To make concise presentation, we do not distinguish the node and the key associated with this node in the remainder of the thesis.

Each key contains the secrete material that is the content of the key and a *key selector* that is used to distinguish the key. As illustrated in Figure 1.2, the key selector consists of: 1) a unique ID that stays the same even if the key content changes and 2) a version and revision field, reflecting update in the keying material. The version number is increased whenever new keying material is sent out by the group manager upon user departure, while the revision number is increased whenever the key is passed through a one-way function. The usage of the version and revision numbers will be explained in the description of the key updating process.

Each user stores his private key, the session key, and a set of KEKs on the path from himself to the root of the key tree. In the example shown in Figure 1.1, user 16 possesses $\{u_{16}, K_s, K_\epsilon, K_1, K_{11}, K_{111}\}$. When a user leaves the service, all his keys need to be updated in order to prevent him from accessing the future communication. Here we use the scheme presented in [18] to demonstrate the key updating process. When user 16 leaves, the KDC generates new keys and conveys new keys to the remaining users through a set of rekeying messages as:

- $\{K_{111}^{new}\}_{u_{15}}$: user 15 acquires K_{111}^{new},

- $\{K_{11}^{new}\}_{K_{111}^{new}}$,$\{K_{11}^{new}\}_{K_{110}^{old}}$: user 13,14,15 acquire K_{11}^{new},
- $\{K_{1}^{new}\}_{K_{11}^{new}}$,$\{K_{1}^{new}\}_{K_{10}^{old}}$: user $9, \cdots, 15$ acquire K_{1}^{new},
- $\{K_{\varepsilon}^{new}\}_{K_{1}^{new}}$,$\{K_{\epsilon}^{new}\}_{K_{0}^{old}}$: user $1, \cdots, 15$ acquire K_{ϵ}^{new},
- $\{K_{s}^{new}\}_{K_{\epsilon}^{new}}$: all remaining users acquire K_{s}^{new},

where the notation x^{old} represents the old version of key x, x^{new} represents the new version of key x, and $\{y\}_x$ represents the key y encrypted by key x. The version numbers are increased for all new keys. This key updating procedure guarantees that all remaining users obtain the new session key and KEKs, while user 16 is unable to acquire the new keys. Since the rekeying messages are transmitted in the multicast channel [17], every user receives all rekeying messages. The session key, KEKs and users' private keys usually have the same length. The communication overhead associated with key updating can be described by *rekeying message size*, defined as the amounts of rekeying messages measured in the unit as the same size as the SK or KEKs. In this example, the rekeying message size is 8 when user 16 leaves the service. It has been shown that the rekeying message size increases linearly with the logarithm of the group size [18].

When a user joins the service, the KDC chooses a leaf position on the key tree to put the joining user. The KDC updates the keys along the path from the new leaf to the root by generating the new keys from the old keys using a one-way function and increasing the revision numbers of the new keys. The joining user obtains the new keys through the unicast channel. Other users in the group will know about the key change when the data packet indicating the increase of the revision numbers first arrives, and compute the new keys using the one-way function. No additional rekeying messages are necessary.

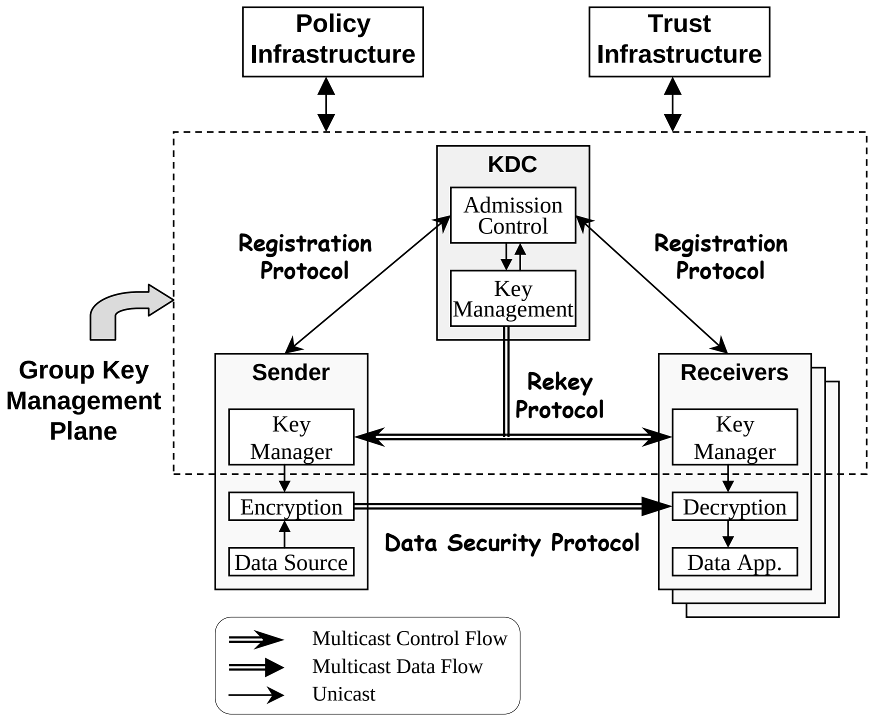


Figure 1.3: Key management architecture in centralized scenario

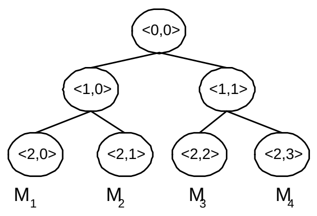
In summary, Figure 1.3 illustrates the high level diagram of centralized access control mechanism for group communications. All users register at the Key Distribution Center (KDC) that generates and distributes keys. With the group session key, secure communication can be established between the sender and the receivers through the *data security protocol*. The KDC knows group membership changes through the *registration protocol* and then updates keys through the *rekey protocol* by transmitting a set of rekeying messages in the multicast channel.

1.3.2 Contributory Key Management

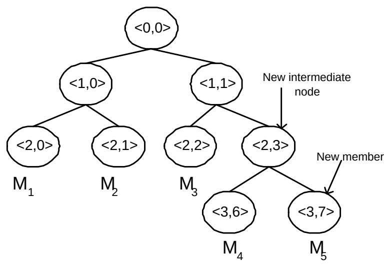
In some scenarios, it is not preferred to rely on a centralized server that arbitrates the establishment of the group key. This might occur in applications where group members do not explicitly trust a single entity, or there are no servers or group members who have sufficient resources to maintain, generate, and distribute keying information. Thus, the distributed solution of the key agreement problem has drawn considerable attention [25–33].

The contributory schemes do not rely on centralized servers. Instead, every group member makes independent contribution and participates the process of group key establishment, and the members' personal keys are not disclosed to any other entities. The early design of contributory key agreements mostly considers the efficiency of key generation for the initial establishment of the group key [25, 27, 28, 34]. Among them, Ingemarsson et al. first introduced a conference key distribution system based on a ring topology [25]. Later, Burmester and Desmedt proposed a key distribution system that takes only three rounds to generate a group key [27]. Steiner et al. extended the two-party Diffie-Hellman (DH) protocol and proposed group Diffie-Hellman protocols GDH.1/2/3 [28,29]. Becker and Willie studied the communication complexity of contributory key agreements and proposed the octopus and 2d-octopus protocols [34]. While achieving efficient initial key establishment, most of these schemes encounter high rekeying complexity upon membership changes. Recent research on key management becomes more aware of the scalability issue in both key establishment and key update for large and dynamic groups. After the tree-based approaches were proposed in the centralized scenario [4, 17], logical tree structure is also used in the contributory setting by Kim et al in their TGDH scheme [31], and by Dondeti et al in their DISEC scheme [32].

Next, we briefly review tree-based contributory key management schemes [31, 32] that use the two-party DH protocol [35] as a basic module. Let A and B denote two entities, K_A denote the private key of A, and K_B denote the private key of B.



(a) a key tree



(b) user join

Figure 1.4: Tree-based contributory key management

Two-party DH protocol establishes a share key between A and B without revealing their private keys as follows. First, A sends the message $\{g^{K_A}\ mod\ p\}$ to B, and B sends the message $\{g^{K_B}\ mod\ p\}$ to A, where g and p, large prime numbers, are exponential base and modular base respectively [35]. Then, A computes $K_{AB} = \left(g^{K_B}\ mod\ p\right)^{K_A}\ mod\ p$; and B computes $K_{AB} = \left(g^{K_A}\ mod\ p\right)^{K_B}\ mod\ p$. Both obtain the shared key K_{AB}.

To establish a shared key among a group of user, the key tree is constructed in a bottom-up fashion. Users are first grouped into pairs and each pair performs a two-party DH to form a sub-group. These sub-groups will again pair up and perform the two-party DH to form larger sub-groups. Continuing in this way, the final group key can be obtained. An example is shown in Figure 1.4(a) with four group members, and member M_i has private key r_i. The group key $K_{\langle 0,0\rangle}$ is computed in two rounds as

1. M_1 and M_2 generate a shared key $K_{\langle 1,0\rangle} = (g^{r_1 r_2} \bmod p)$; and M_3 and M_4 generate a shared key $K_{\langle 1,1\rangle} = (g^{r_3 r_4} \bmod p)$. Then, M_1 and M_2 form a subgroup; and M_3 and M_4 form a subgroup.

2. Two subgroups perform two-party DH protocol and generate the group key

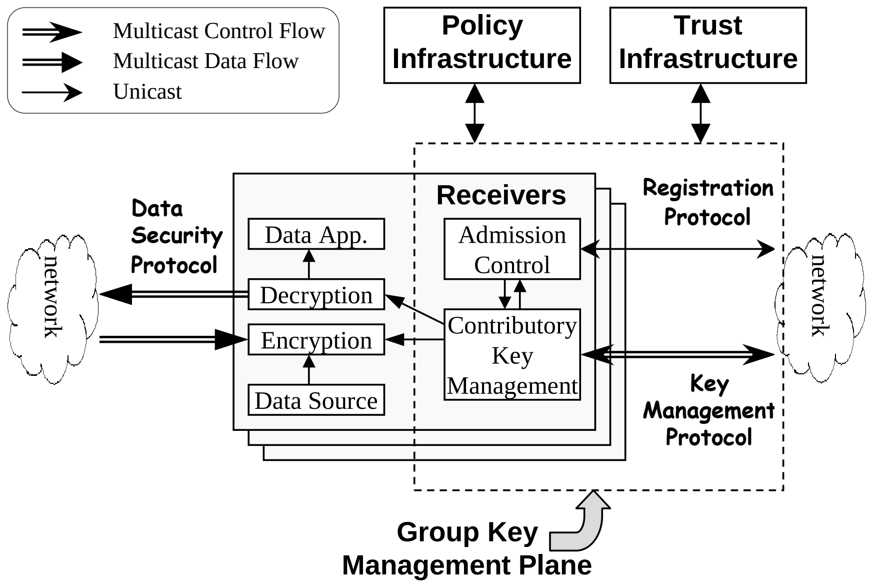


Figure 1.5: Key Management architecture in centralized sceneries

as $K_{\langle 0,0\rangle} = (g^{K_{\langle 1,0\rangle}K_{\langle 1,1\rangle}} \bmod p)$.

In a user join event, the new user will first be paired with an insertion node, which could be either a leaf node or an inner node, to perform a two-party DH. Then all the keys on the path from the insertion node to the tree root are updated recursively. An example is shown in Figure 1.4(b). When member M_5 joins the group, the insertion node is chosen as node $\langle 2,3\rangle$ in Figure 1.4(a), then M_4 and M_5 perform one round of DH to generate a new inner node $\langle 2,3\rangle$ in Figure 1.4(b), followed by the key updates on the path $\langle 2,3\rangle \to \langle 1,1\rangle \to \langle 0,0\rangle$.

Upon a user's departure, the leaving user's node and its parent node will be deleted from the key tree. Its sibling node will assume the position of its parent node. Then all the keys on the path from the leaving user's grandparent node to the tree root are recalculated from the bottom to the top.

Compared with centralized schemes, contributory key management has the advantage of not relying on a single trusted key server, but requires performing computationally expensive cryptographic primitives, such as modular multiplication and exponentiation [36,37]. As a summary, Figure 1.5 illustrates the high level diagram of contributory access control mechanism for group communications.

1.4 Thesis Overview and Contribution

Key management is accomplished either by using a centralized entity that is responsible for distributing keys to users, or by contributory protocols where legitimate members exchange information to agree upon a key. Typical group key management schemes seek to minimize either the amount of rounds needed in establishing the group key, or the size of the rekeying messages. However, those approaches do not factor in the varying requirements of the users, the underlying network, or the application, and are therefore not well suited to provide efficient solutions for all users, for all networks, or for all types of applications. The first two components of this dissertation focus on tailoring access control solutions to wireless networks where users are mobile and the medium is inherently unreliable, and to multimedia applications where the rich properties of the content allow for an improved design of key management. All these scenarios introduce challenges that are not present in conventional key management for generic applications. In order to design better security protocols, it is necessary to look at the security system from an adversarial point-of-view. The third component of this dissertation addresses a security concern in popular key management schemes and proposes a framework to immunize the key management protocols. Next, we introduce these three components individually.

Topology-aware key management in wireless networks

To achieve forward and backward security, *rekeying messages* are sent to group members when there are users joining or leaving the multicast group. In applications where there are many users and frequent additions or deletions to the group membership, key management can introduce a significant communication burden. rekeying messages must be delivered reliably because the loss of rekeying messages results in severe performance degradation [3]. If a user loses one key, he will not be able to access multicast content encrypted by this key and may not be able to acquire future keys from future rekeying messages either. Further, in real-time multicast applications the rekeying messages should also be delivered in a timely manner so that users receive the rekeying messages before the new key takes effect. These reasons alone motivate the need for building communication-efficient key management schemes. In wireless multicast scenarios, however, the need is even more pronounced since bandwidth is limited and data typically experience a higher transmission error rate than in conventional environments.

In the first part of this dissertation, we propose a method for designing a centralized multicast key management tree for a group of users in a cellular network. Traditional tree-based multicast key management schemes do not consider the effect of the network topology upon the delivery of the rekeying messages, and therefore waste network resources by sending rekeying messages to users who do not need them. We address this issue by proposing to match the key management tree to the network topology, thereby localizing the delivery of the rekeying messages and reducing the communication costs. In mobile environments, the user will subscribe to a multicast service under an initial host agent, and through the course of his service undergo *handoff* to different base stations. We will discuss issues arising from user relocation and present a handoff scheme that is suitable for topology-matching key management. In addition, we prove that optimizing the proposed key tree is equivalent to optimizing a set of independent smaller-scale subtrees. This significantly reduces the complexity of the tree design. A tree structure that can easily adapt to changes in the number of users and a tree generation algorithm that considers the heterogeneity of the network will also be introduced.

Hierarchical Group Access Control

Existing key management schemes, such as in [4, 16–33], address the access control issues in a single multicast session. They focus on establishing and updating keys with dynamic membership and provide all group members the same level of access privilege. That is, the users who possess the decryption keys have the full access to the content, and the users who do not have the decryption keys cannot interpret the data. In practice, many group applications contain multiple related data streams and have the members with various access privileges. These applications prevail in various scenarios.

- Multimedia applications distributing data in multi-layer coding format [38]. For example, in a video broadcast, users with a normal TV receiver can receive the normal format, while others with HDTV receivers can receive both the normal format and the extra information needed to achieve HDTV resolution.

- Multicast programs containing several related services, such as weather, news, traffic and stock quote.

- Communications in hierarchically managed organizations, such as military group communications where participants have various access authorization.

Since group members subscribe to different data steams, or possibly multiple of them, it is necessary to develop access control mechanism that supports the multi-level access privilege, which shall be referred to as the *hierarchical group access control*.

The access control issue for each data stream can be managed separately using existing key management schemes . However, this leads to inefficient use of keys and does not scale well when the number of data streams increases. In the second part of this dissertation, we develop a *multi-group key management* scheme that addresses the generalized hierarchical group access control problem. Particularly, we design an *integrated key graph* that maintains the keying material for all members with different access privileges and incorporates new functionalities that are not present in conventional multicast key management, such as the user relocation on the key graph. The proposed multi-group key management scheme achieves forward and backward secrecy [31] when users (1) join the group communication with certain access privilege; (2) leave the group; and (3) add or drop the subscription of one or several data streams (change access privilege). The idea of the integrated key graph can be used in both centralized and contributory environments. Compared with using single-session access control solutions, such as a variety of tree-based key management scheme [18, 31], the proposed scheme reduces the usage of the communication, computation and storage overhead, and is scalable when the number of access levels increases.

Securing Dynamic Group Membership Information

Key management is employed to prevent unauthorized access to multicast content. We discovered, however, the rekeying process associated with multicast key management can disclose information about the dynamics of the group membership to both insiders and outsiders. We collectively refer to group dynamics information (GDI) as information describing the dynamic membership of a group application, such as the number of users in the multicast group as a function of time, and the number of users who join or leave the service during a time interval. The leakage of GDI from the rekeying process can lead to serious security and privacy problems. For example, in a commercial multicast program, the service provider performs group management and has the knowledge of GDI. However, it is highly undesirable to disclose instant detailed dynamic membership information to competitors, who could develop effective competition strategies by analyzing the statistical behavior of the audience. Another example is a military group communication scenario, where GDI represents the number of soldiers on the battlefield and the number of soldiers moving into or out of certain areas. In this situation, the valid group members, i.e. regular soldiers, may only be entitled to obtain general information through the secure group communication, but not to acquire GDI. Further, leaking GDI to outsiders, most likely to the enemies, can be devastating. In the wireless scenario, the need for studying GDI leakage and developing leakage-immune multicast key management schemes is even more pronounced because the broadcast nature of the wireless media enables anyone within the broadcast range to observe the encrypted data.

In the third part of this dissertation, we demonstrate that the key management schemes can reveal the GDI easily and propose a framework of protecting GDI from inside and outside attackers. We have developed two effective strategies to attack and *steal* information about the membership dynamics from the tree-based centralized schemes [3, 4, 16–19] that employ tree hierarchy for the maintenance of keying material. These strategies involve exploiting the format of rekeying messages and estimating GDI directly from the size of the rekeying messages. We also developed an anti-attack method that is fully compatible with the existing key management schemes. By utilizing batch rekeying [39] and introducing phantom users, the proposed anti-attack method aims to minimize the mutual information between the rekeying process observed by the attackers and the true group dynamics. Various aspects of the proposed anti-attack scheme, such as the communication overhead and the leakage of GDI, are evaluated based on the data obtained from MBone sessions. The analysis on other non-tree based schemes is also provided. In contributory key management, each group member need to be aware of other group members in order to establish the shared group key. Thus, the task of protecting GDI is more difficult than it is in the centralized scenario. We provide qualitative analysis on the vulnerability of various contributory schemes and techniques that can be used to protect GDI in the distributed environments.

The rest of the dissertation is organized as follows. The topology-matching key management scheme for wireless networks is discussed in Chapter 2. The key-graph based hierarchical group access control is presented in Chapter 3. In Chapter 4, we discuss attack and protection technologies for dynamic group membership information. Conclusion and future work is in Chapter 5.

Chapter 2

Topology-aware Key Management for Wireless Networks

2.1 Introduction

There has been significant advancements in building a global wireless infrastructure that will free users from the confines of static communication networks. Users will be able to access the Internet from anywhere at anytime. As wireless connections become ubiquitous, consumers will desire to have multicast applications running on their mobile devices. In order to meet such a demand, there has been increasing research efforts in the area of wireless multicast [40–42].

In wireless networks, where bandwidth is limited and transmission error rate is high, the design of key management schemes need to consider the transmission of the rekeying messages in order to ensure reliable key distribution and reduce the communication burden associated with key management. Previous key management schemes focus entirely on generating the rekeying messages, but they neglect the issues of the delivery of the rekeying messages and do not consider the underlying network topology.

In this Chapter, we propose a topology-aware centralized key management scheme for multicast applications in a cellular network. By matching the key management tree to the network topology and localizing the delivery of rekeying messages, we can significantly reduce the communication burden associated with rekeying. In Section 2.2, we introduce the concept of matching the key tree to the network topology and motivate the reduction in the communication cost associated with rekeying. In mobile environments, the user will subscribe to a multicast service under an initial host agent, and through the course of his service undergo *handoff* to different base stations. In Section 2.3, we discuss issues arising from user relocation and present a handoff scheme that is suitable for topology-matching key management. In Section 2.4, we analyze the effect that matching the key tree to topology has upon the communication overhead. We then address the complexity of designing the key management tree in Section 2.5 by proving that optimizing the proposed key tree is equivalent to optimizing a set of independent smaller-scale subtrees. This significantly reduces the complexity of the tree design. We describe, in Section 2.6, a tree structure that can easily adapt to changes in the number of users and a tree generation algorithm that considers the heterogeneity of the network. We then describe a procedure to build the key tree and determine the parameters that optimize the tree. Finally, simulation results are presented in Section 2.7.

2.2 Topology-Matching Key Management Tree

In this section, we introduce the benefits of matching the key tree to the network topology. We outline a procedure to design the key management tree and define the cost functions that we use in the rest of the chapter for measuring the communication burden associated with key updating.

Let us revisit the example of tree-based centralized key management in Section 4.2.3 (see Figure 1.1). As user 16 leaves the multicast service, all of his keys are updated through a set of rekeying messages. It is seen that most rekeying messages are only useful to a subset of users, who are always neighbors on the key management tree. In fact, the first rekeying message is only useful to user 15, the second rekeying message is only useful to users 13,14,15, the third rekeying message is useful to users $9, 10, \cdots, 15$, and the fourth and fifth rekeying messages are useful to all users. Therefore, rekeying messages do not have to be sent to every user in the multicast group.

We propose to exploit this observation in designing a key management tree. Our key management tree will match the network topology in such a way that the neighbors on the key tree are also physical neighbors on the network. By delivering the rekeying messages only to the users who need them, we may take advantage of the fact that the key tree matches the network topology, and localize the delivery of rekeying messages to small regions of the network. This lessens the amount of traffic crossing portions of the network that do not have users who need to be rekeyed. In order to accomplish this, it is necessary to have the assistance of entities that would control the rekeying message transmission, such as the base stations in cellular wireless networks.

A cellular network model, as depicted in Figure 2.1 and proposed in [43], consists of mobile users, base stations (BS) and supervisor hosts (SH). The SHs administrate the BSs and handle most of the routing and protocol details for mobile users. The service provider, the SHs, and the BSs are connected through high-speed wired

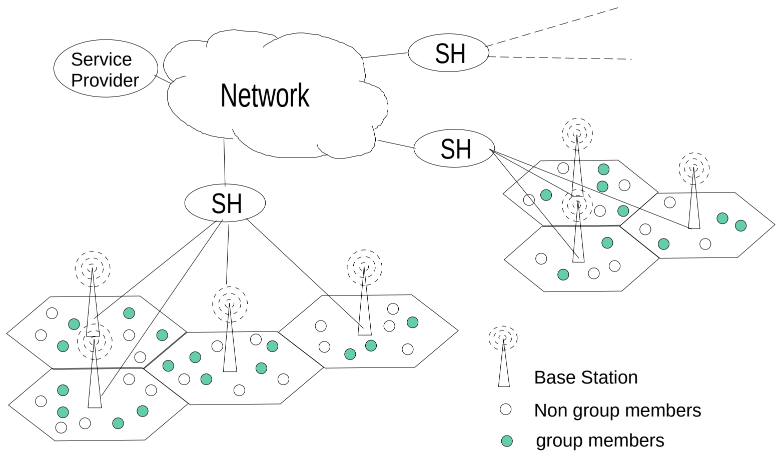


Figure 2.1: A cellular wireless network model

connections, while the BSs and the mobile users are connected through wireless channels. In this work, the SHs can represent any entity that administers BSs, such as the region servers presented in [44] and radio network controllers (RNCs) in 3G networks [45]. In cellular wireless networks, multicast communication can be implemented efficiently by exploiting the inherent broadcasting nature of the wireless media [46–48]. In this case, multicast data is first routed to the BSs using multicast routing techniques designed for wireline networks [2], and then broadcast by the BSs to mobile users.

If we assume that both the SHs and the BSs can determine whether the rekeying messages are useful for the users under them, then the cellular wireless network has the capability of sending messages to a subset of users. In particular, the SHs multicast a rekeying message to their BSs if and only if the message is useful to one or several of their BSs, and the BSs broadcast the rekeying message to their users if and only if the message is useful to the users under them. The information needed to identify whether a SH or BS needs a rekeying message can be sent in the rekeying message header. We shall not consider the size of this overhead

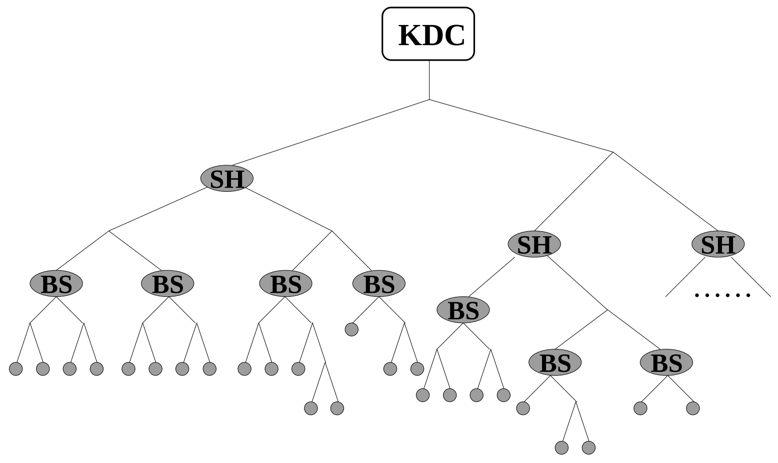


Figure 2.2: A Topology -matching key management tree

information in our calculation since this overhead is typically small compared to the size of the actual rekeying messages, and is implementation-dependent. Hence, when the key tree matches the network topology, we can localize the delivery of rekeying messages.

We design a key management tree that matches the network topology in three steps:

- Step 1: Design a subtree for the users under each BS. These subtrees are referred to as *user subtrees*.
- Step 2: Design subtrees that govern the key hierarchy between the BSs and the SH. These subtrees are referred to as *BS subtrees*.
- Step 3: Design a subtree that governs the key hierarchy between the SH and the KDC. This subtree is referred to as the *SH subtree*.

The combined key management tree is called a Topology-Matching Key Management (TMKM) tree. Figure 2.2 illustrates a TMKM tree for the network topology shown in Figure 2.1. Traditional key management trees, such as those in [16–19], are independent of the network topology, and we call them Topology Independent Key Management (TIKM) trees. When using a TIKM tree, the users are scattered all over the network, and therefore it is not possible to localize the delivery of rekeying messages.

We study the communication burden of the rekeying messages in the wired portion and in the wireless portion of the network separately. Under each SH, the *wireline-message-size* is defined as the total size of the rekeying messages multicast by the SHs to the BSs, and the *wireless-message-size* is defined as the total size of the rekeying messages broadcast by the BSs. The message size is measured in units whose bit length is the same size as the key length. In this work, we assume that the network connection between the KDC and the SHs has ample bandwidth resource and experience very low error rate. Thus, the wireline-message-size does not include the communication overhead between the KDC and the SHs.

Let S_1^l denote the wireline-message-size under the l^{th} SH and S_2^l denote the wireless-message-size under the l^{th} SH, where $l = 1, 2, \cdots, n_{sh}$ and n_{sh} is the total number of SHs. For example, when the length of the session key and KEKs is 128 bits each, if a 256 bit long rekeying message is multicast by the l^{th} SH and then broadcast by 3 BSs under the l^{th} SH, then $S_1^l = 2$ and $S_2^l = 6$. Assuming that users do not leave simultaneously, then the rekeying wireline cost, C_{wire}, the rekeying wireless cost $C_{wireless}$, and the total rekeying cost C_T, are defined as:

$$\begin{aligned} C_{wire} &= \sum_{l=1}^{n_{sh}} \alpha_1^l E[S_1^l] \ ; \quad C_{wireless} = \sum_{l=1}^{n_{sh}} \alpha_2^l E[S_2^l] \\ C_T &= \gamma \cdot C_{wireless} + (1-\gamma) \cdot C_{wire} \end{aligned} \tag{2.1}$$

where $E[.]$ indicates expectation over the statistics governing the user joining and leaving behavior. Here, $0 \leq \gamma \leq 1$ is the *wireless weight*, which represents the importance of considering the wireless cost, and $\{\alpha_1^l\}$ and $\{\alpha_2^l\}$ are the sets of weight factors that describe the importance of considering the wireline-message-size and wireless-messages-size under the l^{th} SH respectively. When SHs administrate areas with similar physical network structure and channel conditions, we can approximate $\{\alpha_1^l\}$ and $\{\alpha_2^l\}$ by 1. In addition, we define the *combined-message-size* as $S_T^l = \gamma \cdot S_2^l \alpha_2^l + (1-\gamma) \cdot S_1^l \alpha_1^l$. Thus, C_T can also be expressed as $C_T = \sum_{l=1}^{n_{sh}} E[S_T^l]$.

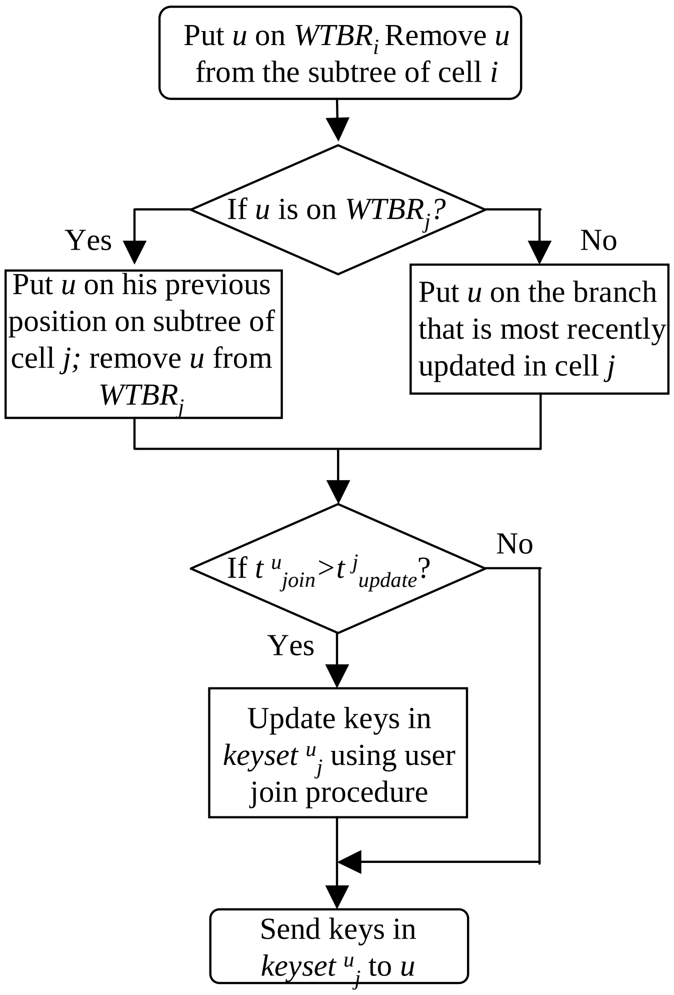
For a given wireless weight γ, $\{\alpha_1^l\}$, and $\{\alpha_2^l\}$, both the TMKM and TIKM trees should be designed to minimize the total communication cost, C_T.

2.3 Handoff Schemes for TMKM Tree

In mobile environments, the user will subscribe to a multicast service under an initial host agent, and through the course of his service move to different cells and undergo *handoff* to different base stations. Although the user has moved, he still maintains his subscription to the multicast group. Since the TMKM tree depends on the network topology, the physical location of a user affects the user's position on the key management tree. When a user moves from one cell to another cell, the user needs to be relocated on the TMKM tree. In this section, we propose an efficient handoff scheme for our TMKM trees. In this context, the expression *handoff scheme* will only refer to the process of relocating a user on the key tree.

One solution to the handoff problem is to treat the moving user as if he departs the service from the cell that he is leaving from and then rejoins the service in the cell that he has moved to. This scheme, referred to as the *simple handoff scheme*, is not practical for mobile networks with frequent handoffs since rekeying messages are sent whenever handoffs occur.

During handoff, if a user remains subscribed to the multicast group, it is not necessary to remove the user from the cell where he previously stayed. Allowing a



Put *u* on *WTBR*$_i$ Remove *u* from the subtree of cell *i*
If *u* is on *WTBR*$_j$?
Yes
No
Put *u* on his previous position on subtree of cell *j*; remove *u* from *WTBR*$_j$
Put *u* on the branch that is most recently updated in cell *j*
If $t^u_{join} > t^j_{update}$?
No
Yes
Update keys in *keyset*^u_j using user join procedure
Send keys in *keyset*^u_j to *u*

Figure 2.3: Key update process when user u moves from cell i to cell j

mobile user to have more than one set of valid keys while he stays in the service does not compromise the requirements of access control, as long as all of the keys that he possesses are updated when he finally leaves the service. In order to trace both the users' handoff behavior and the key updating process, we employ a wait-to-be-removed (WTBR) list for each cell. The WTBR list of the cell i, denoted by $WTBR_i$, contains the users who (1) possess a set of valid keys on the user subtree of cell i and (2) are currently in the service but not in cell i. These WTBR lists are maintained by the KDC.

Let t^i_{update} denote the time of the last key update that occurs due to a departure occurring in cell i, and let t^u_{join} denote the time when the user u first joins the service. In addition, we define $keyset^u_i$ to be the set of keys possessed by the user u while he is in cell i. We propose an *efficient handoff scheme* that is illustrated in Figure 2.3 and Figure 2.4, as:

- When user u moves from cell i to cell j,

    1. Put u on the WTBR list of cell i, i.e. $WTBR_i$, and remove him from the user subtree of cell i.
    2. If u has been in cell j before and is on $WTBR_j$, put u back on the branch of the subtree that he previously belonged to and remove him from $WTBR_j$. If u is not on $WTBR_j$, put u on the most recently updated branch on the user subtree of cell j. We note that the set of keys associated with u's new position, $keyset^u_j$, was updated at time t^j_{update}.
    3. If $t^u_{join} > t^j_{update}$, the keys in $keyset^u_j$ are updated using the procedure for user join described in [18]. If $t^u_{join} \leq t^j_{update}$, the keys do not need to be updated.
    4. The keys in $keyset^u_j$ are sent to u through unicast.

  The purpose of step 3 is to prevent u from taking advantage of the handoff process to access the communication that occurred before he joined. To see this, let u join the service at $t^u_{join} = t_0$ in cell i, and then immediately move to cell j. After relocation, user u obtains keys in $keyset^u_j$ that is updated at time $t^j_{update} = t_0 - \Delta$, where Δ is a positive number. In this case, if we do not update the keys in $keyset^u_j$ and u has recorded the communication in cell

j before joining, u will be able to decrypt the multicast content transmitted in $[t_0 - \Delta, t_0)$, during which time he is not a valid group member.

- When user u leaves the multicast service from cell j:

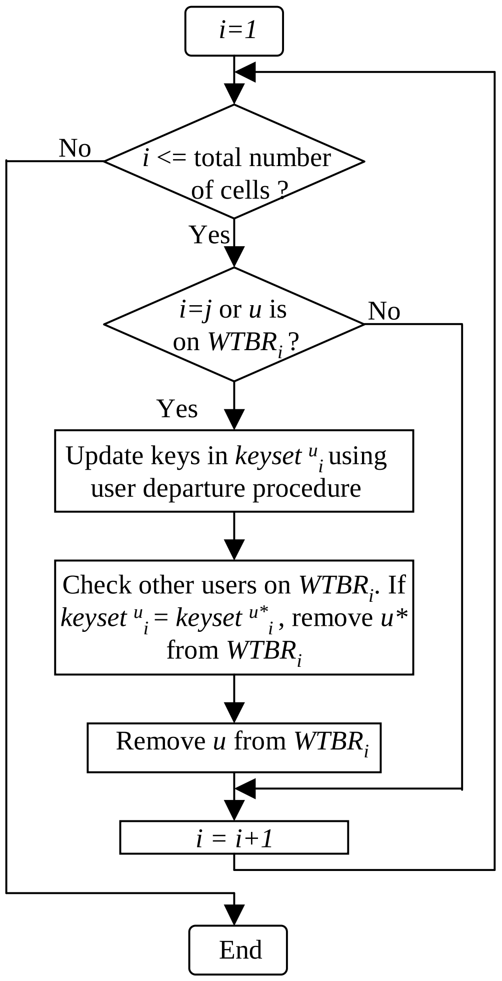
  1. The keys that are processed by u and still valid should be updated. In particular, the keys in $keyset_j^u$ and $\{keyset_i^u : WTBR_i \text{ contains } u\}$ are updated using the procedure for user departure in [18].

  2. Check other users on the WTBR lists that contain u. If u and another user u^* are both on $WTBR_i$, and $keyset_i^u = keyset_i^{u^*}$, remove u^* from $WTBR_i$. It is noted that u^* is removed from $WTBR_i$ when u^* does not have valid keys associated with cell i any more. Step 2 does not require extra rekeying messages.

  3. Remove u from all WTBR lists.

Thus, a user will be removed from the WTBR lists not only when he leaves the service, but also when other users who share the same keys leave the service. Compared with the simple handoff scheme, the efficient handoff scheme can reduce the key updating caused by user relocation because the number of cells that need to update keys is smaller than the number of cells that a user has ever visited.

When the key tree matches with the network topology, handoffs result in users' relocation on the key tree, which inevitably introduce extra cost to the task of key management. In this work, we assume that the KDC has significant computation and storage resources and do not investigate the cost for the KDC to maintain and update the WTBR lists. We will focus on the extra communication cost due to the fact that more than one set of keys may need to be updated for a departure user when handoffs exist.



i=1
i <= total number of cells ?
No
Yes
i=j or u is on WTBR_i ?
No
Yes
Update keys in keyset ^u_i using user departure procedure
Check other users on WTBR_i. If keyset ^u_i = keyset ^u*_i , remove u* from WTBR_i
Remove u from WTBR_i
i = i+1
End

Figure 2.4: Key update process when user u leaves the service from cell j

2.4 Performance Analysis

Matching the key management tree with the network topology has two contrasting effects on the rekeying message communication cost. First, the cost of sending one rekeying message is reduced because only a subset of the BSs broadcast the message. Second, the number of rekeying messages may increase due to handoffs. In this section, we analyze these two effects and investigate the influence that user mobility and the wireless weight have upon the performance of the TMKM scheme.

To simplify the analysis, we assume that the system has a^{L_0} SHs, each SH administrates a^{L_1} BSs, and each BS has a^{L_2} users, where $a \geq 2$, L_0, L_1 and L_2 are positive integers. We also assume that the SHs administer areas with similar network structure and conditions. Therefore, $\{\alpha_1^l\}$ and $\{\alpha_2^l\}$ are approximated by 1. The user subtrees, BS subtrees, and SH subtree are designed as balanced trees with degree a and level L_2, L_1, and L_0, respectively. For fair comparison, the TIKM tree is also designed as an *a-ary* balanced tree with $(L_0 + L_1 + L_2)$ levels. In this work, the level of a tree is defined as the maximum number of nodes on the path from a leaf node to the root excluding the leaf node. Since the SHs are usually in charge of large areas, the probability of a user moving between SHs during a multicast service is much smaller than the probability of handoffs that are under one SH. In this analysis, we assume that there are no SH level handoffs. For the present computation, we only calculate the communication cost caused by *one* departure user based on the rekeying procedure described in [3,17,18].

As illustrated by the example in Section 2.2, rekeying messages with size $(a \cdot L)$ need to be transmitted when one user leaves from a balanced key tree with degree a and level L. When using the TIKM tree, rekeying messages with size $a(L_0+L_1+L_2)$ are transmitted under a^{L_0} SHs and broadcast by $a^{L_0+L_1}$ BSs. Therefore, when one user leaves the service, wireline-message-size, denoted by $\widetilde{C}_w^{tikm}$, and the wireless-message-size, denoted by $\widetilde{C}_{wl}^{tikm}$, are computed as

$$\widetilde{C}_w^{tikm} = (aL_0 + aL_1 + aL_2)a^{L_0} \tag{2.2}$$

$$\widetilde{C}_{wl}^{tikm} = (aL_0 + aL_1 + aL_2)a^{L_0+L_1}. \tag{2.3}$$

The performance of the TMKM tree is affected by the user handoff behavior. We define the random variable I as the number of WTBR lists that contain the departing member when he leaves the service. We also introduce the function

$B(b, i, a)$ that describes the number of intermediate KEKs that need to be updated. $B(b, i, a)$ is equivalent to the expected number of occupied boxes when putting i items in b boxes with repetition, where each box can have at most a items. A box is called occupied when one or more items are put into the box. The detailed calculation of $B(b, i, a)$ is given in Appendix A.

When one user leaves the service and he is on $I = i$ WTBR lists, we can show that:

- We need to update $(i \cdot L_2)$ keys on user subtrees. Thus, rekeying messages with total size $(iaL_2 - 1)$ are transmitted under one SH and broadcast by a single BS.
- We need to update $B(a^{L_1-m}, i, a^m)$ KEKs on the level $(L_1 - m)$ of the BS subtree. Thus, messages with size $aB(a^{L_1-m}, i, a^m)$ are transmitted under one SH and broadcast by a^m BSs. Here, $m = 1, \cdots, L_1$, and the level 0 of a tree is just the root.
- We need to update (a^t) KEKs on the level $(L_0 - t)$ of the SH subtree. Thus, messages with size (a^{t+1}) are sent under (a^t) SHs and broadcast by $(a^{L_1} \cdot a^t)$ BSs. Here, $t = 1, 2, \cdots, L_0$.
- In addition, we need one message to update the session key K_s. This message is sent to all a^{L_0} SHs and $a^{L_0+L_1}$ BSs.

Therefore, when the departing user belongs to i WTBR lists, the expected value of the wireline-message-size, denoted by $C_w^{tmkm}(i)$, and the expected value of the wireless-message-size, denoted by $C_{wl}^{tmkm}(i)$, are computed as

$$C_w^{tmkm}(i) = iaL_2 + \sum_{m=1}^{L_1} aB(a^{L_1-m}, i, a^m) + \sum_{t=1}^{L_0} a^{t+1} \tag{2.4}$$

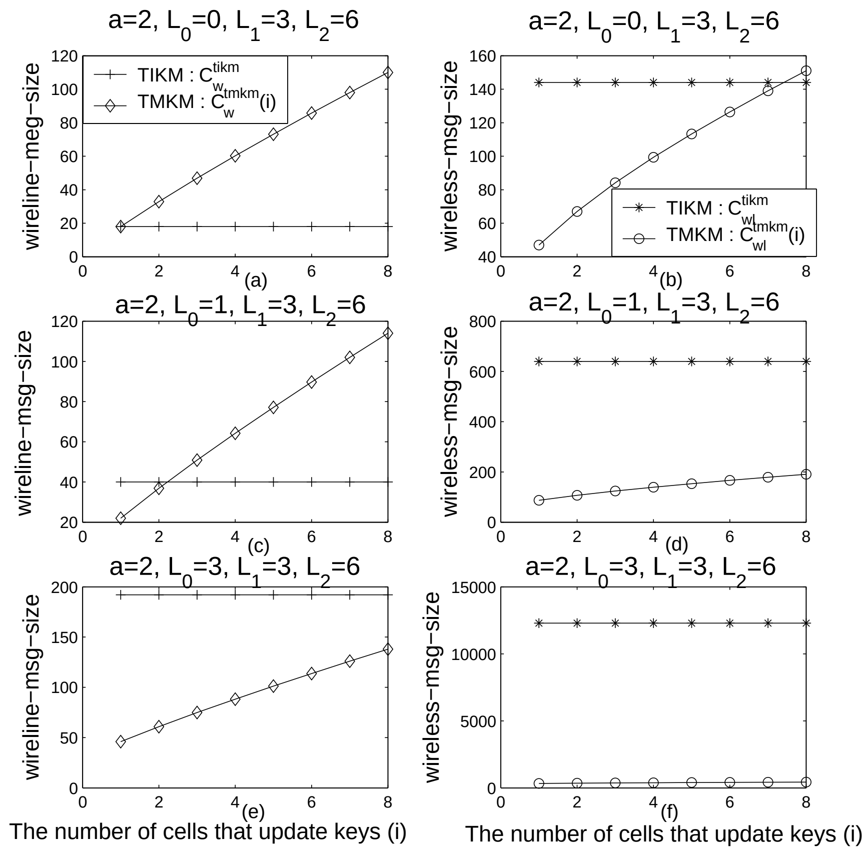


Figure 2.5: Comparison of the wireless cost and the wireline cost for one user departure

$$
\begin{aligned}
C_{wl}^{tmkm}(i) &= iaL_2 - 1 + \sum_{m=1}^{L_1} a^{m+1} B(a^{L_1-m}, i, a^m) \\
&\quad + a^{L_1} \sum_{t=1}^{L_0} a^{t+1} + a^{L_0+L_1}. \qquad (2.5)
\end{aligned}
$$

The performance of the TIKM tree and the TMKM tree can be compared by examining the values of $\widetilde{C}_w^{tikm}$ and $C_w^{tmkm}(i)$, $\widetilde{C}_{wl}^{tikm}$ and $C_{wl}^{tmkm}(i)$. In Figure 2.5, these values are plotted for different i and L_0, when the other parameters are fixed as $a = 2$, $L_1 = 3$, and $L_2 = 6$. Since the TIKM tree is not affected by handoffs, $\widetilde{C}_w^{tikm}$ and $\widetilde{C}_w^{tikm}$ are constant. Figure 2.5(a) and Figure 2.5(b) show the wireline-message-size and wireless-message-size respectively, when the system has only one SH. Figure 2.5(c) and Figure 2.5(d) show the corresponding curves for 2 SHs, while Figure 2.5(e) and Figure 2.5(f) depict the corresponding curves for systems with 8 SHs. We observe that:

- Both $C_w^{tmkm}(i)$ and $C_{wl}^{tmkm}(i)$ are increasing functions of i.
- The TMKM tree always reduces the wireless-message-size, and this advantage becomes larger when the system contains more SHs.
- For systems containing only one SH, i.e. $L_0 = 0$, the TMKM trees introduce larger wireline-message-size than TIKM trees due to the handoff effects. When there are multiple SHs, the TMKM scheme can take advantage of the fact that some SHs do not need to transmit rekeying messages to their BSs, and can reduce the wireline-message-size when i is small. It should be noted that the wireline cost will be larger than that given in (2.4) if there are SH-level handoffs.

Since TMKM trees reduce the wireless-message-size more effectively than reducing the wireline message size, a larger wireless weight γ leads to an improved advantage of TMKM trees over TIKM trees. Using large γ is a reasonable scenario since the wireless portion of the network usually experiences a higher error rate and has less available bandwidth when compared to the wireline portion, which makes the wireless cost the major concern in many realistic systems. In addition, the communication cost of the TMKM tree increases with the number of cells that need to update keys when a user leaves. Therefore, when handoffs are less likely to happen, the TMKM tree has larger advantage over the TIKM tree.

Scalability is another important performance measure of key management schemes [3]. We define $N = a^{L_0}$ as the number of SHs. When $N \to \infty$, the scalability properties can be easily obtained from (2.2)-(2.5), and are summarized in Table 2.1. Both Figure 2.5 and Table 2.1 demonstrate that the communication cost of TMKM trees scales better than that of TIKM trees when more SHs participate in the multicast.

	wireline-message-size	wireless-message-size
TIKM	$\sim aN\log_a N$	$\sim a^{L_1+1}N\log_a N$
TMKM	$\sim a^2\log_a N$	$\sim a^{L_1+2}\log_a N$

Table 2.1: Scalability comparison between TMKM and TIKM trees when the number of SHs(N)$\to \infty$.

2.5 Separability of the Optimization Problem

The TMKM tree consists of user-subtrees, BS-subtrees, and SH-subtrees. In this section, we show that optimizing the entire TMKM tree is equivalent to optimizing those subtrees individually. This is desirable since optimizing the subtrees separately reduces the dimension of the search space for optimal tree parameters and significantly reduces the complexity of tree design.

In this work, we assume that the users under the same SH have the same joining, departure and mobility behavior. Thus, the user subtrees under the same SH have the same structure. It is easy to verify that the main results in this section still hold in scenarios where the dynamic behavior of the users varies under different BSs. However, for the discussion in this chapter, we will restrict our attention to the case where the dynamic behavior of the users between different

BSs is identical. In addition, we assume that the number of participating SHs and BSs do not change during the multicast service. In order to make the presentation more concise, we introduce the notation $D_{k,l}$ to represent the situation where k users are under the l^{th} SH and one of these users leaves the service.

As discussed in Section 2.2, the total communication cost, C_T, is expressed as

$$C_T = \sum_{l=1}^{n_{sh}} E[S_T^l]. \tag{2.6}$$

Based on the definition of S_T^l, one can see that

$$E[S_T^l] = \sum_k p^l(k) G^l(k) E^l(k), \tag{2.7}$$

where

$p^l(k)$:	pmf of the number of users under the l^{th} SH,
$G^l(k)$:	probability that a user leaves from the l^{th} SH given that k users are under the l^{th} SH,
$E^l(k)$:	the expected value of the combined-message-size given the condition $D_{k,l}$.

When a user leaves, the keys that need to be updated are divided into three categories: (1) the keys on the user subtrees, (2) the keys on the BS subtrees, and (3) the keys on the SH subtree. Under the condition $D_{k,l}$, let $A_1^l(k)$, $A_2^l(k)$ and A_3^l denote the expected value of the combined-message-size under the l^{th} SH resulting from updating the keys on the user-subtrees, BS-subtrees and SH-subtrees, respectively. We note that A_3^l is not a function of k when there are no SH-level handoffs, and that $E^l(k) = A_1^l(k) + A_2^l(k) + A_3^l$. Then, (2.6) becomes

$$C_T = \sum_{l=1}^{n_{sh}} \left(\sum_k p^l(k) G^l(k) A_1^l(k) + \sum_k p^l(k) G^l(k) A_2^l(k) + A_3^l \cdot \left(\sum_k p^l(k) G^l(k) \right) \right).$$

We observe that the structure of the user-subtrees only affects $A_1^l(k)$, the structure of the BS-subtrees only affects $A_2^l(k)$, and the structure of the SH-subtrees only affects A_3^l. Therefore, for the TMKM tree, the user-subtrees, BS-subtrees and SH subtree can be designed and optimized separately. Particularly, the user-subtrees under the l^{th} SH should be designed to minimize $\sum_k p^l(k)G^l(k)A_1^l(k)$, the BS subtree under the l^{th} SH should be designed to minimize $\sum_k p^l(k)G^l(k)A_2^l(k)$, and the SH subtree should be designed to minimize $\sum_{l=1}^{n_{sh}} A_3^l \cdot \Big(\sum_k p^l(k)G^l(k)\Big)$.

2.6 Design of the TMKM Tree

Key management schemes are closely related to the *key management architecture*, which describes the entities in the network that perform key management [3]. In cellular wireless networks, the BSs are not trusted to perform key management because they can be easily tampered with [43]. The SHs are able to perform key management if they are trusted and have the necessary computation and storage capabilities. The trustiness of the SHs depends on both the business model and the protection on the SHs. Based on whether SHs perform key management, the systems can be classified into two categories:

- In the first category, each SH performs key management for a subset of the group members who reside in the region where this SH is in charge. Each SH can be looked at as a local key distribution center. Without loss of generality, since the SHs are independent and may even adopt different key management schemes, we can study systems containing only one SH, which we shall refer to as *one-SH systems*.

- In the second category, SHs do not perform key management. Instead, there is a KDC that manages keys for all users. This KDC can be the service provider or a trusted third party. The systems containing many SHs are referred to as *multiple-SH systems*.

In one-SH systems, the TMKM tree consists of user-subtrees and a BS subtree. In multiple-SH systems, the TMKM tree consists of user-subtrees, BS-subtrees and a SH subtree.

In this section, we introduce a model describing the joining and leaving behavior of the users, and a flexible tree structure that can be used to design the user and BS subtrees. We then examine the optimization of the user and BS subtrees and the design of the SH subtree.

2.6.1 Dynamic membership model

Mlisten [49] is a tool that can collect the join/leave times for multicast group members in MBone sessions. Using this tool, [50] [51] studied the characteristics of the membership dynamics of MBone multicast sessions and showed that the user arrival process can be modeled as Poisson and the membership duration of short sessions (that usually last several hours) is accurately modeled using an exponential distribution while the membership duration of long sessions (that usually last several days) is accurately modeled using the Zipf distribution [52]. Based on the population model of short MBone sessions, we made the following assumptions on the membership dynamics:

1. Under the l^{th} SH, the user's arrival process is Poisson with rate λ_l and the service duration is governed by an exponential random variable with mean $1/\mu_l$, where $l = 1, 2, \cdots, n_{sh}$.

2. A user's joining and leaving behavior is independent of other users.

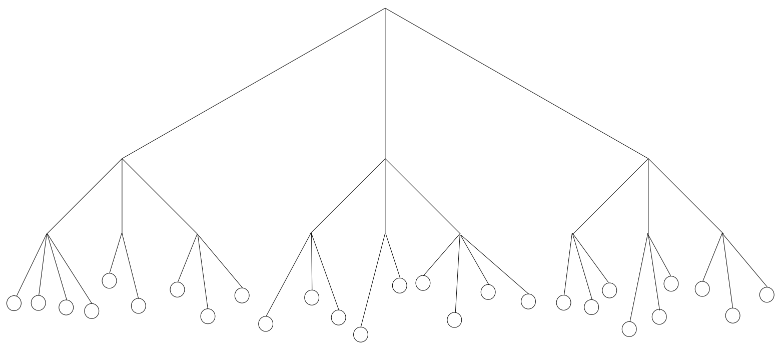


Figure 2.6: ALX tree

Based on the first assumption, the number of users under the l^{th} SH is a Poisson random variable with rate θ_l, i.e. $p^l(k) = \frac{\theta_l^k}{k!}e^{-\theta_l}$, where $\theta_l = \lambda_l/\mu_l$ [53]. In addition, it can be shown that $G^l(k)$ approximately equals to $k \cdot \mu_l$. It is noted that the second assumption is reasonable in some types of multicast services, such as periodic news multicast, while it may not be correct for services such as a scheduled pay-per-view multicast, where different users are related with each other through watching the same content.

In this work, we use this Poisson arrival and exponential service duration model to optimize the TMKM tree. In Section 2.7, we will use simulations to demonstrate that the performance of the TMKM tree is not sensitive to users' statistical membership models.

2.6.2 ALX tree structure

The TMKM scheme matches the key tree to the network topology by decomposing the key tree into user subtrees, BS subtrees, and SH subtrees. The TMKM scheme does not have constraints on the specific structure of these subtrees. In this section, we propose a tree structure that is capable of handling membership additions, deletions, or relocations with minimal changes to the tree's structure.

As illustrated in Figure 2.6 and parameterized by the triple $(a, L, \mathbf{x})$, this $(a, L, \mathbf{x})$-logical tree has $L + 1$ levels. The upper L levels, which comprise a full balanced subtree with degree a, are fixed during the multicast service. The users are represented by the leaf nodes on the $(L + 1)^{st}$ level. We use a vector $\mathbf{x}$ to describe the $(L + 1)^{st}$ level, where x_i is the number of users attached to the i^{th} node of the L^{th} level, and $i = 1, 2, \cdots, a^L$. In the example shown in Figure 2.6, $\mathbf{x} = [4, 2, 3, 3, 2, 4, 3, 3, 3]$, $a = 3$ and $L = 3$. We will refer to this tree structure as the ALX tree.

When using the ALX tree, the joining user is always put on the branch with the smallest value of x_i. The maximum number of users on an ALX tree is not restricted. When a user leaves, the average rekeying message size is $(\frac{k}{a^L} - 1 + aL)$, where k is the number of users on the ALX tree. When the user's arrival process is Poisson with rate λ, and the service time is an exponential random variable with mean $1/\mu$, the probability that a user leaves the key tree is approximately $k \cdot \mu$, and the pmf of k is $p(k) = \frac{\theta^k}{k!}e^{-\theta}$, where $\theta = \lambda/\mu$. The performance of the ALX tree is evaluated by the expected value of the rekeying message size, denoted by C_{alx}, and is calculated as:

$$C_{alx} = \sum_{k=1}^{\infty} p(k) \cdot k \cdot \mu \cdot (\frac{k}{a^L} - 1 + aL), \tag{2.8}$$

It follows that the optimization problem of the ALX tree can be formulated as:

$$\widetilde{C}_{alx} = \min_{a>1, L>0} C_{alx}. \tag{2.9}$$

Balanced trees whose degree is pre-determined, such as binary and trinary trees, are widely used to design key trees [3, 18]. Next, we compare the ALX tree structure with balanced trees that have a pre-defined degree, which we refer to as fixed-degree trees in this section.

Adding or removing a user from balanced fixed-degree trees often requires splitting or merging nodes. For example, when a new user is added to the key tree shown in Figure 1.1, one leaf node must be split to accommodate the joining user. In this case, a new KEK is created and must be transmitted to at least one existing user. When using the ALX tree structure, however, no new KEKs are created during membership changes. We know that updating existing KEKs for user join can be achieved without sending any rekeying messages, as suggested in [18], because existing users can update KEKs using one-way functions after being informed of the need to update their keys. Therefore, the ALX tree structure allows for a key updating operation that does not require sending any rekeying messages during user joins. In addition, the ALX tree introduces minimal change to the tree structure with dynamic membership and therefore is easy to implement and analyze.

On the other hand, the ALX tree is optimized over the distribution of the group size. If we take individual snapshots of the system when the group size is very small or large, the ALX tree may not perform as well as fixed degree trees that adjust themselves according to the group size. However, we will derive the performance lower bound for fixed degree trees and then demonstrate that the cost for ALX trees, $\widetilde{C}_{alx}$, is in fact very close to this lower bound. Similar to (2.8), the expected rekeying message size when using a tree with fixed degree n, denoted by $C_{fix}(n)$, is calculated as:

$$C_{fix}(n) = \sum_{k=1}^{\infty} p(k) \cdot k \cdot \mu(n - 1 + n \cdot (P - 1)) \ ,$$

where P is the average length of branches for a tree with k leaves and degree n. It is well known that P equals the expected codeword length of a source code containing k symbols with equal probability. The bounds on P are known to be

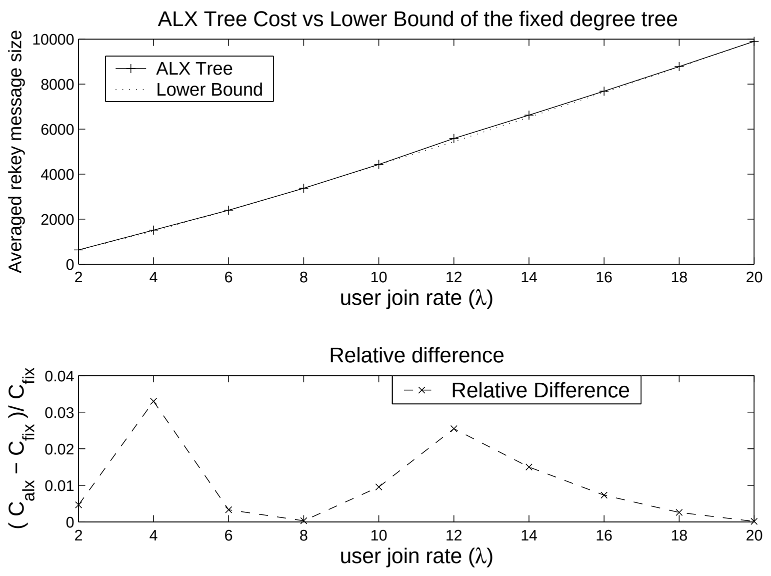


Figure 2.7: Comparison between the ALX tree performance and the lower bound for different user joining rates

$\log_n(k) \le P < \log_n(k) + 1$ [54]. Therefore,

$$C_{fix}(n) > \sum_{k=1}^{\infty} p(k) \cdot k \cdot \mu \cdot (n \log_n(k) - 1). \tag{2.10}$$

Based on (2.10), the performance lower bound for the fixed degree trees is given by

$$\widetilde{C}_{fix} = \min_n \sum_{k=1}^{\infty} p(k) \cdot k \cdot \mu \cdot (n \log_n(k) - 1). \tag{2.11}$$

It is noted that no fixed degree trees can reach this lower bound. In fact, $\widetilde{C}_{fix}$ would be achieved if and only if we could (1) reorganize the tree immediately after user join or departure in such a way that the rekeying message size for the next user join/leave operation is minimized; and (2) reorganize the tree without adding any extra communication cost. However, reorganizing trees, such as splitting or merging nodes, requires sending extra keying information to users. These above two conditions can never be achieved simultaneously.

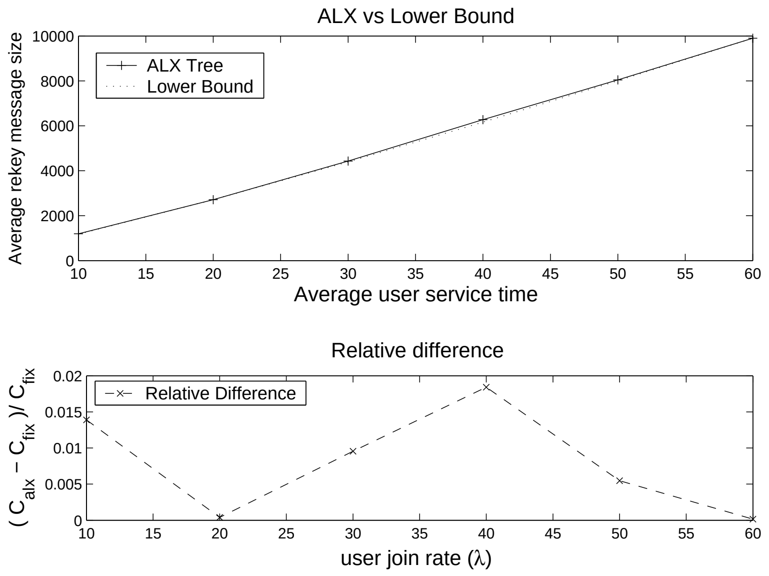


Figure 2.8: Comparison between the ALX tree performance and the lower bound for different average service duration

The lower bound in (2.11) is used as a reference for evaluating the performance of the ALX tree. In Figure 2.7, $\widetilde{C}_{fix}$ and $\widetilde{C}_{alx}$ are compared for different user joining rates, λ. In Figure 2.8, $\widetilde{C}_{fix}$ and $\widetilde{C}_{alx}$ are compared for different average service duration, $1/\mu$. We observe that the relative difference between the lower bound and the performance of the ALX tree is less than 3.5%.

The ALX tree has the advantage of maintaining tree structure as user join and leaves, while its performance is very close to the lower bound of fixed degree trees. Although the ALX tree is not the optimal solution amongst all possible tree structures, its practical nature makes the ALX tree an ideal candidate for designing the user and BS subtrees.

2.6.3 User subtree design

The user subtrees are designed as ALX trees. Under the l^{th} SH, the optimal tree parameters, a and L, solve

$$\min_{a,L} \sum_k p^l(k) G^l(k) A_1^l(k), \tag{2.12}$$

where a and L are positive integers and $G^l(k) \approx k\mu_l$. Let $T_w^u(k,i)$ and $T_{wl}^u(k,i)$ respectively represent the expected value of the wireline-message-size and wireless-message-size caused by updating keys on the user subtrees, given that k users are under the l^{th} SH, one of them leaves and he is on i WTBR lists. We can show that

$$T_w^u(k,i) = T_{wl}^u(k,i) = (\frac{k/n_{bs}^l}{a^L} - 1 + aL)i.$$

Then, $A_1^l(k)$ is computed as

$$\begin{aligned} A_1^l(k) &= \sum_{i=1}^{n_{bs}^l} p_h^l(i)(\alpha_2^l \gamma T_{wl}^u(k,i) + \alpha_1^l(1-\gamma) T_w^u(k,i)) \\ &= (\alpha_2^l \gamma + \alpha_1^l(1-\gamma))(\frac{k/n_{bs}^l}{a^L} - 1 + aL) E[I^l], \end{aligned} \tag{2.13}$$

where $E[I^l] = \sum_{i=1}^{n_{bs}^l} p_h^l(i) i$, and α_1^l and α_2^l are defined in Section 2.2. By substituting (2.13) into (2.12), the optimization problem for the user-subtrees under the l^{th} SH is

$$\min_{a,L} \sum_k k \cdot p^l(k) \cdot (\frac{k/n_{bs}^l}{a^L} - 1 + aL). \tag{2.14}$$

The optimum a and L can be obtained by searching the space of possible a and L values.

2.6.4 BS subtree design

We also design BS subtrees as ALX trees. We denote the degree and the level of a BS subtree by a_{bs} and L_{bs}, respectively. Let $T_w^b(k,i)$ and $T_{wl}^b(k,i)$ respectively denote the expected value of the wireline-message-size and wireless-message-size caused by key updating on the BS subtree under the l^{th} SH given the condition $D_{k,l}$ and the condition that the departing member is on i WTBR lists. We can show that:

$$T_w^b(k,i) = s \cdot B({a_{bs}}^{L_{bs}}, i, s) + \sum_{m=1}^{L_{bs}} a_{bs} \cdot B({a_{bs}}^{L_{bs}-m}, i, s \cdot a_{bs}^m) \tag{2.15}$$

$$\begin{aligned} T_{wl}^b(k,i) \approx\ & s^2 \cdot B({a_{bs}}^{L_{bs}}, i, s) \\ & + \sum_{m=1}^{L_{bs}} a_{bs} \cdot {a_{bs}}^m \cdot s \cdot B({a_{bs}}^{L_{bs}-m}, i, s \cdot a_{bs}^m), \end{aligned} \tag{2.16}$$

where $s = \frac{n_{bs}^l}{a_{bs}^{L_{bs}}}$. Equation (2.15) and (2.16) are derived based on the following intermediate results:

- On average, $B({a_{bs}}^{L_{bs}-m}, i, s \cdot a_{bs}^m)$ keys need to be updated on level $(L_{bs} - m)$ of the BS subtree.
- To update one KEK at level L_{bs}, the average message size is (s) and these messages are broadcast to an average of (s) BSs. To update one KEK at level $(L_{bs} - m), m > 0$, the message size is (a_{bs}) and these messages are broadcast by (a_{bs}^m) BSs.

From the definition of A_2^l and using both (2.15) and (2.16), we can see that

$$\begin{aligned} A_2^l &= \sum_{i=1}^{n_{bs}^l} p_h^l(i)(\alpha_2^l \gamma T_{wl}^b(k,i) + \alpha_1^l(1-\gamma)T_w^b(k,i)) \\ &= \sum_{i=1}^{n_{bs}^l} p_h^l(i) \Bigg(B({a_{bs}}^{L_{bs}}, i, s) \left(s^2 \alpha_2^l \gamma + s\alpha_1^l(1-\gamma)\right) \\ &+ \sum_{m=1}^{L_{bs}} B({a_{bs}}^{L_{bs}-m}, i, s a_{bs}^m) a_{bs} \left(a_{bs}^m s \alpha_2^l \gamma + \alpha_1^l(1-\gamma)\right) \Bigg), \end{aligned} \tag{2.17}$$

where n_{bs}^l is the number of BSs under the l^{th} SH. In practice, it is difficult to obtain an analytic expression for $p_h^l(i)$ that depends on the statistical behavior of the users during membership joins and departures, as well as their mobility behavior and how handoffs are addressed. Thus, we introduce random variable $\widetilde{I}^l$, which is the number of cells that a leaving user has ever visited. Obviously, $\widetilde{I}^l \geq I^l$. The pmf of $\widetilde{I}^l$, denoted by $\widetilde{p}_h^l(i)$, can be derived from user mobility behavior and the distribution of the service duration, as described in Appendix B. Let $\widetilde{A}_2^l$ denote the right hand side value in (2.17) when replacing $p_h^l(i)$ by $\widetilde{p}_h^l(i)$. We can show that $\widetilde{A}_2^l$ is an upper bound of A_2^l. We notice that $\widetilde{A}_2^l$ is not a function of k.

As discussed in Section 2.5, the parameters of the BS subtree under the l^{th} SH should be chosen to minimize $\sum_k p^l(k)G^l(k)A_2^l$. Since $G^l(k)$ is not a function of a_{bs} and L_{bs}, minimizing $\sum_k p^l(k)G^l(k)A_2^l$ is equivalent to minimizing A_2^l. Due to the unavailability of $p_h^l(i)$, we choose the parameter of the BS subtrees under the l^{th} SH that minimize the upper bound of A_2^l, as

$$\min_{a_{bs}>1, L_{bs}>0} \widetilde{A}_2^l. \tag{2.18}$$

2.6.5 SH subtree design

In a typical cellular network, each SH administrates a large area where both the user dynamics and the network conditions may differ significantly from the areas administered by other SHs. The heterogeneity among the SHs should be considered in designing the SH subtree. Due to SH heterogeneity, the ALX tree structure, which treats every leaf equally, is not an appropriate tree structure to build the SH subtree. Instead, the SH heterogeneity may be addressed by building a tree where the SHs have varying path lengths from the root to their leaf node. In this section, we will first formulate the SH subtree design problem and then provide a

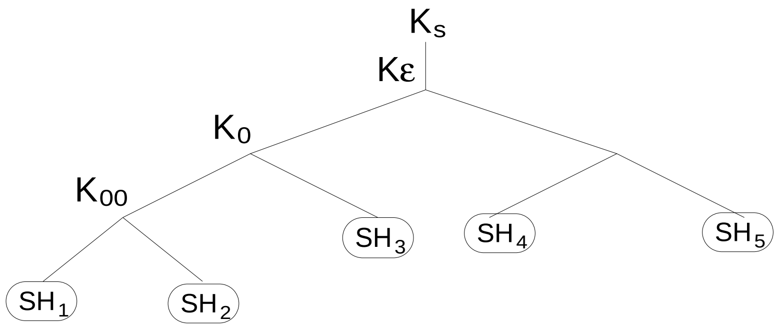


Figure 2.9: An example of the SH subtree

sub-optimal tree generation procedure.

The root of the SH subtree is the KDC, and the leaves are the SHs. The design goal is to minimize the third term in (2.8), which shall be denoted by C_{sh} and is

$$C_{sh} = \sum_{l=1}^{n_{sh}} q_l \cdot A_3^l, \tag{2.19}$$

where $q_l = \sum_k p^l(k) G^l(k)$. Let β_l denote the communication cost of transmitting one rekeying message to all the users under the l^{th} SH. Based on the definition of α_1^l and α_2^l in Section 2.2, it is easy to show that $\beta_l = (1-\gamma)\alpha_1^l + \gamma n_{bs}^l \alpha_2^l$.

The value of A_3^l can be calculated directly from β_l where $l = 1, 2, \cdots, n_{sh}$. In the simple example demonstrated in Figure 2.9, when a user under SH_1 leaves the multicast service, K_{00}, K_0, K_ϵ and K_s, need to be updated. The communication cost of updating K_{00} is $2(\beta_1 + \beta_2)$. The communication cost of updating K_0 is $2(\beta_1 + \beta_2 + \beta_3)$. The communication cost of updating K_ϵ is $2(\beta_1+\beta_2+\beta_3+\beta_4+\beta_5)$. Since the communication cost of updating K_s does not depend on SH subtree structure, it is not counted in the total communication cost. Then, we have:

$$A_3^1 = 2(\beta_1 + \beta_2) + 2(\beta_1 + \beta_2 + \beta_3) + 2(\beta_1 + \beta_2 + \beta_3 + \beta_4 + \beta_5).$$

The goal of the SH subtree design is to find a tree structure that minimizes C_{sh} given β_l and q_l. However, it is very difficult to do so based on (2.19). Thus, we

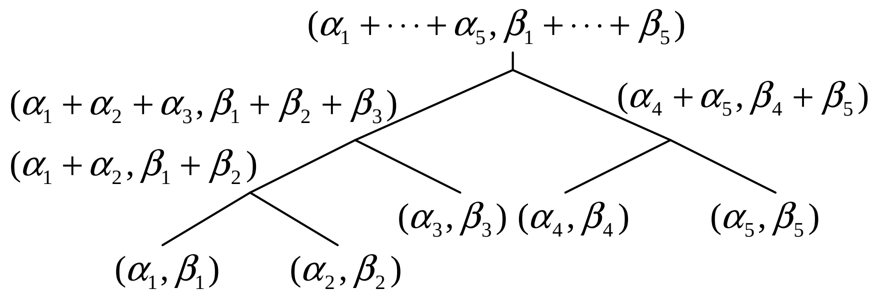

$(\alpha_1 + \cdots + \alpha_5, \beta_1 + \cdots + \beta_5)$
$(\alpha_1 + \alpha_2 + \alpha_3, \beta_1 + \beta_2 + \beta_3)$
$(\alpha_4 + \alpha_5, \beta_4 + \beta_5)$
$(\alpha_1 + \alpha_2, \beta_1 + \beta_2)$
(α_3, β_3)
(α_4, β_4)
(α_5, β_5)
(α_1, β_1)
(α_2, β_2)

Figure 2.10: The cost pairs on the SH subtree

compute C_{sh} in a different way.

We assume that the SH subtree has the fixed degree n. We shall assign a *cost pair*, which is a pair of positive numbers, to each node on the tree as follows. The cost pair of the leaf node that represents the l^{th} SH is (q_l, β_l). The cost pair of the intermediate nodes are the element-wise summation of their children nodes' cost pairs, as illustrated in Figure 2.10. The cost pairs of all intermediate nodes are represented by (x_m, y_m), where $m = 1, 2, \cdots, M$, and M is the total number of intermediate nodes on the tree. Then, C_{sh} can be calculated as

$$C_{sh} = n \sum_{m=1}^{M} x_m \cdot y_m. \tag{2.20}$$

It is easy to verify that (2.20) is equivalent to (2.19). Based on (2.20), we propose a tree construction method for $n = 2$ as

1. Label all the leaf nodes using their cost pairs, and mark them to be active nodes.

2. Choose two active nodes, (x_i, y_i) and (x_j, y_j), such that $(x_i + x_j) \cdot (y_i + y_j)$ is minimized among all possible pairs of active nodes. Mark those two nodes to be inactive and merge them to generate a new active node with the cost pair $(x_i + x_j, y_i + y_j)$.

3. Repeat step 2 until there is only one active node left.

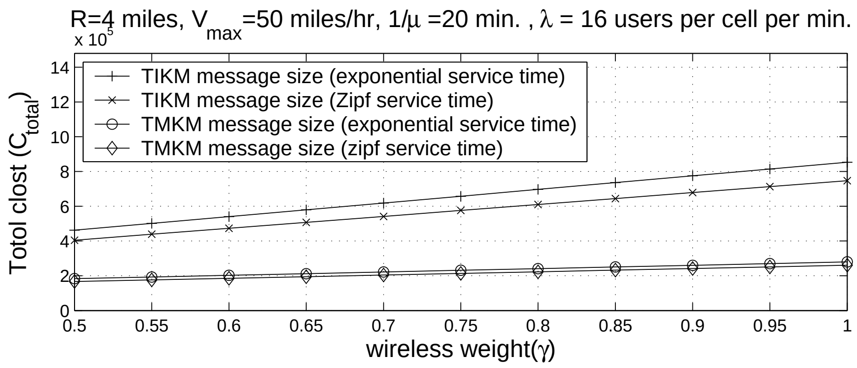
This method, which we call the greedy-SH subtree-design (GSHD) algorithm, can be easily extended to $n > 2$ cases. We can prove that the GSHD algorithm produces the optimal solution when $\beta_1 = \beta_2 = \cdots \beta_{n_{sh}}$, but is not optimal in general cases. Since the optimization problem for the SH-subtree is non-linear, combinatorial, and even does not have a closed expression for the objective function, we do not seek the optimal SH subtree structure in this work. In Section 2.7, we will compare the performance of the GSHD algorithm and the optimal solution obtained by exhaustive search.

2.7 Simulation Results

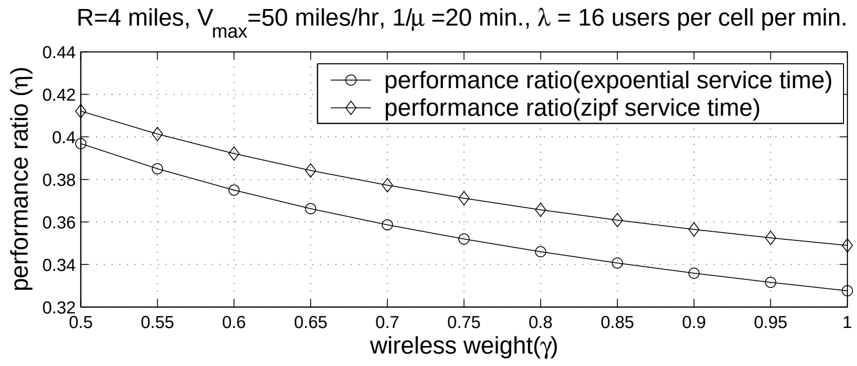
2.7.1 One-SH systems

We first compare the performance of the TMKM tree and the TIKM tree in one-SH systems by both analysis and simulations. Similar to [55, 56], we employ a homogeneous cellular network that consists of 12 concatenated cells, and wrap the cell pattern to avoid edge effects. We use the mobility model proposed in [57], where R denotes the radius of the cells, and V_{max} denotes the maximum speed of the mobile users. Since the wireless connection usually experiences a high transmission error rate and the number of users under one BS is larger than the number of BSs, the wireless communication cost of the multicast communication is assigned a larger weight than the wireline communication cost, i.e. $\gamma > 0.5$.

For the purpose of fair comparison, the TIKM tree is designed as an ALX tree, which is optimized for the statistics of the number of participating users. The wireline cost of the TIKM tree, denoted by C^{tikm}_{wire}, is computed using (2.8),



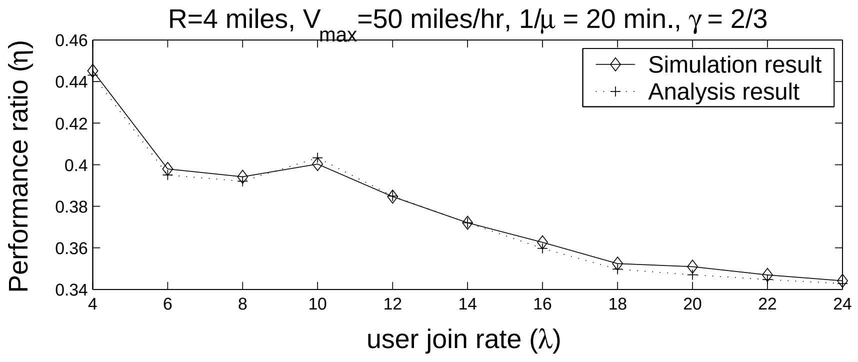
(a)



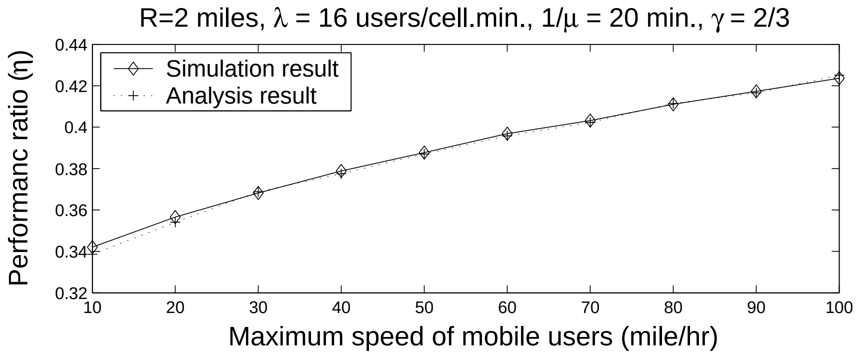
(b)

Figure 2.11: (a) The total message size as a function of the wireless weight; (b) Performance ratio as a function of the wireless weight

where $p(k)$ denotes the pmf of the number of users in the multicast service. The wireless cost of the TIKM tree is computed as $C_{wireless}^{tikm} = n_{bs} C_{wire}^{tikm}$, where n_{bs} is the total number of BSs. In one-SH systems, the total communication cost is $C_T^{tikm} = \gamma C_{wireless}^{tikm} + (1-\gamma) C_{wire}^{tikm}$. We define the *performance ratio* η as the total communication cost of the TMKM tree divided by the total communication cost of the TIKM tree, i.e. $\eta = C_T^{tmkm}/C_T^{tikm}$. When η is less than 1, the TMKM tree has smaller communication cost than the TIKM tree, and smaller η indicates an improved advantage that the TMKM tree has over the TIKM tree.



(a)



(b)

Figure 2.12: (a) Performance ratio for different user join rate; (b) Performance ratio for different users' maximum speed.

Figure 2.11(a) shows the total communication cost of the TMKM tree and the TIKM tree for different wireless weights (γ), when the cellular cells have a radius of 4 miles, the maximum mobile speed is 50 miles/hour, and the user joining rate is 16 users per minute per cell. The corresponding performance ratio is shown in Figure 2.11(b). In this simulation, two models are used to describe users' join/departure behavior. The first one, representing short sessions, uses a Poisson arrival and exponential service time duration model. The second one, representing long sessions, uses a Poisson arrival and Zipf service time duration model. The users stay in the service for an average of 20 minutes in both cases. Three observations are made. First, the communication cost of the TMKM tree is always less than 42% of the communication cost of the TIKM tree. Second, the performance ratio η is smaller for larger γ, which supports the argument in Section 2.4 that the advantage of the TMKM tree is larger when more emphasis is placed on the wireless cost. Third, when the wireless transmission is the bottleneck of the system, i.e. $\gamma = 1$, the TMKM tree can reduce the communication burden by as much as 65%, i.e. $\eta = 35\%$. In addition, two models yield similar results, which indicates that the performance of the TMKM is not sensitive to the models. In the remainder of this section, we adopt the short session model.

Figure 2.12(a) shows both the analysis and the simulation results of η for different user join rates (λ) when the radius of the cellular cells is 4 miles, the maximum mobile speed is 50 miles per hour, the average service time ($1/\mu$) is 20 minutes, and $\gamma = 2/3$. Since the exact expression for the pmf of I^l is not available, to calculate analytical results, we use an empirically estimated pmf of I^l, which is obtained from simulations with the same user join/departure and mobility models. We can see that the advantage of the TMKM tree is larger when the system contains more users. This property can be verified by studying the cost functions derived in the previous sections. In Figure 2.12(b), the performance ratio is shown for different V_{max} when the user joining rate is 16 users per minute per cell. The performance ratio is an increasing function of V_{max} when other parameters are fixed since handoffs occur more frequently as users move faster.

2.7.2 Multiple-SH systems

As discussed in Section 2.6, when the system contains multiple SHs that do not perform key management, the design of the TMKM tree should consider the topology

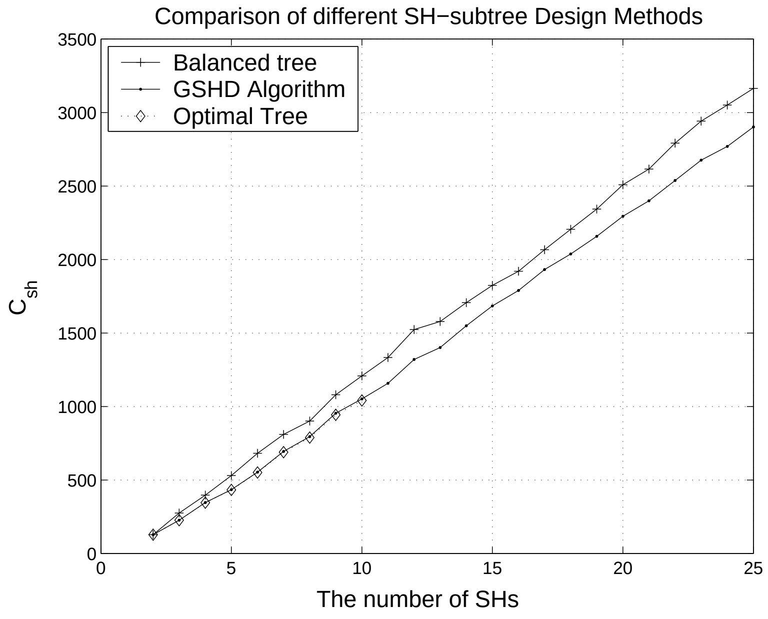


Figure 2.13: Comparison among SH subtree design methods

of the SHs.

SH subtree design methods

In this section, we compare the GSHD algorithm with the optimal tree obtained by exhaustive search, and with a balanced tree that treats each SHs equally and represents traditional key management schemes. We assume that half of the $\{\beta_l\}$ are uniformly distributed between 1 and 20, which represent rural areas, and the other half of $\{\beta_l\}$ uniformly distributed between 101 and 120, which represent metropolitan areas. We also assume that q_l, which is defined in Section 2.6.5 and represents the probability of a user leaving, is proportional to β_l, where $l = 1, 2, \cdots, n_{sh}$. Here, $\{q_l\}$ are normalized such that $\sum q_l = 1$. In Figure 2.13, the communication cost caused by updating keys on SH-subtrees, C_{sh}, is shown when using different SH subtrees. Results are averaged over 500 realizations. Since

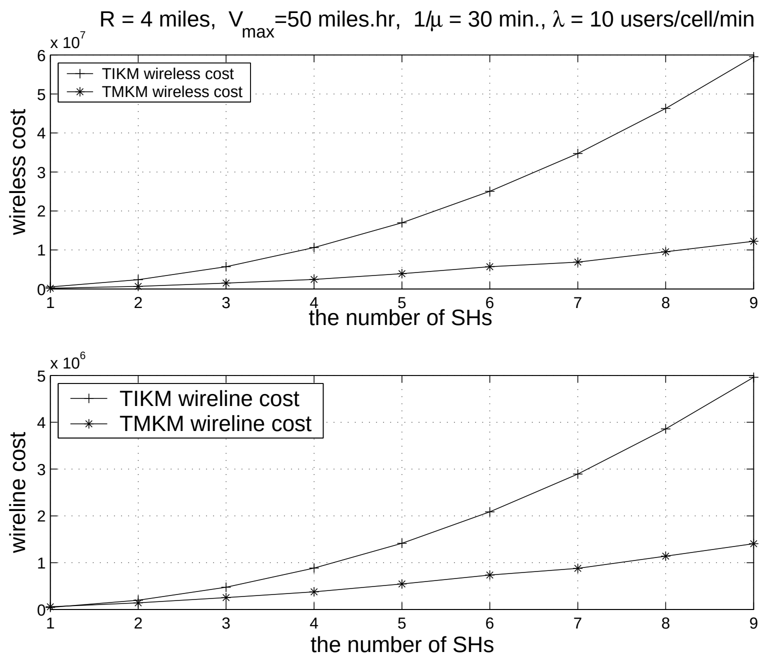


Figure 2.14: Performance comparison in multiple-SH systems with identical SHs

exhaustive search is very computationally expensive, it is only done for 10 and fewer SHs. The simulation results indicate that the performance of the GSHD is very close to optimal. Compared with the balanced tree, GSHD algorithm reduces the communication cost contributed by the SH subtree by up to 18%.

Performance of TMKM trees and TIKM trees in multiple-SH systems

For the TMKM trees in multiple-SH systems, we designed the user-subtrees and BS-subtrees as ALX trees, while the SH-subtrees were constructed using the GSHD algorithm. We simulated a multiple-SH system where each SH administers 12 concatenated identical cells. The SH-subtrees are constructed as binary trees. We first study a simple case where the user statistics and network conditions are identical under all SHs. In this case, α_1^l's and α_2^l's are set to be 1. The radius of the cells is $R = 4$ miles, the maximum velocity is $V_{max} = 50$ miles/hr, and we also

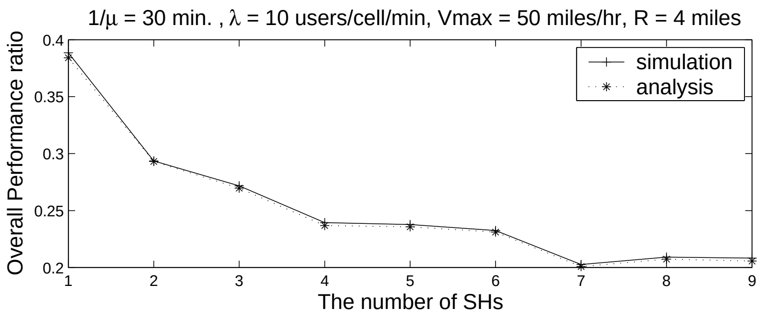


Figure 2.15: Performance comparison in multiple-SH systems with non-identical SHs

choose $\mu_l = 1/30$ and $\lambda_l = 10$ for all SHs.

In Figure 2.14, the wireless cost and the wireline cost of the TMKM trees and the TIKM tree are shown for different quantities of participating SHs. We observed that the TMKM trees have both smaller wireless cost and smaller wireline costs than the TIKM trees when the number of SHs are equal or greater than 2, and the advantages of the TMKM trees are more significant when the system contains more SHs, which verifies the analysis in Section 2.4. In addition, the corresponding performance ratio is drawn in Figure 2.15 for $\gamma = 2/3$. In this system, the communication cost of the TMKM trees can be as low as 20% of the communication cost of the TIKM trees. This indicates an 80% reduction in the communication cost.

A more complicated system containing 5 SHs with different user joining rates was also simulated. In this scenario, the λ_l values for the five SHs were set to 5, 10, 15, 20 and 25 respectively, and $R = 4$ miles, $V_{max} = 50$ miles/hr, and $\mu_l = 1/20$ for all SHs. The TMKM tree structure is shown in Figure 2.16. The TIKM tree is simply an ALX tree with degree 3 and level 6. In this system, the wireless cost of the TMKM tree is 21.8% of that of the TIKM tree, and the wireline cost of the

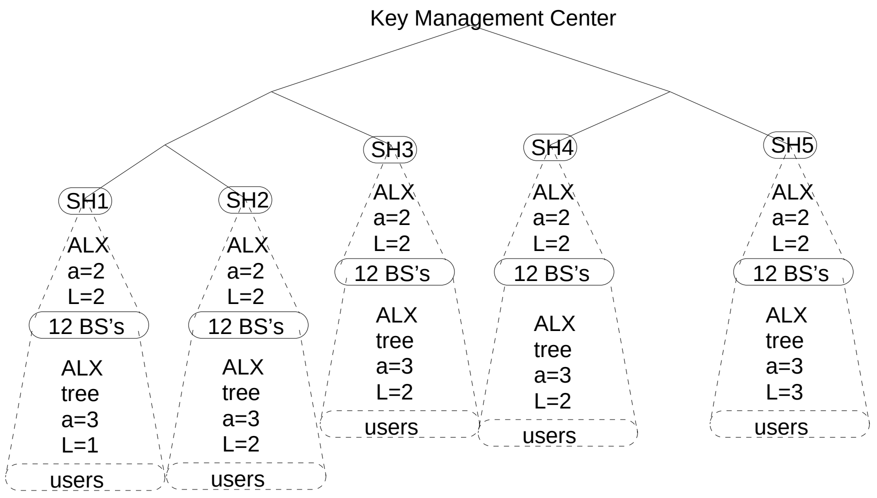


Figure 2.16: A TMKM tree containing 5 SHs

TMKM tree is 34.0% of that of the TIKM tree. When the wireless weight γ is set to 2/3, the TMKM tree reduced total communication cost by 74%.

In this chapter, we described a topology-aware multicast key management scheme for mobile wireless environment. Compared with traditional tree-based key management schemes that are independent of network topology, the proposed TMKM scheme achieved a significant reduction in the communication burden associated with rekeying. The proposed key tree consists of user-subtrees, BS-subtrees and SH-subtrees. We proved that the problem of optimizing the communication cost for the TMKM tree is separable and can be solved by optimizing each of those subtrees separately. This property greatly reduced the complexity in key tree design. The ALX tree structure, which easily adapts to changes in the number of users, was introduced to build user-subtrees and BS-subtrees. The GSHD algorithm, which considers the network heterogeneity where the SHs administer areas with varying network conditions, was introduced to build the SH subtree. An efficient handoff scheme was introduced to address the consequences that user mobility has upon the TMKM tree. Both simulations and analysis demonstrated that the proposed TMKM scheme can significantly reduces the communication cost. In addition, the communication cost of the TMKM tree scales better than that of topology-independent trees as the number of participating SHs increases.

Chapter 3

Hierarchical Group Access Control

3.1 Introduction

Many group communications involve multiple data streams and group members with various access privileges. This type of group communication prevails in multimedia applications that often distribute data in multi-layer coding format [38], in multicast applications containing several related services, and in hierarchical managed organizations.

Traditional multicast key management schemes are not designed to handle key management issues associated with multiple services occurring concurrently that have correlated memberships. New key management scheme need to be designed that exploit the overlap in the memberships of different data streams, while incorporating new functionalities that are not present in conventional multicast key management. Specifically, it is necessary to introduce new rekeying events that allow users to subscribe or cancel membership to some layers while maintaining their membership to others.

As discussed in Section 1.4, the straightforward approach to achieve hierarchical access control is to associate a separate key tree with each data stream. The advantage of such a scheme lies primarily in the simplicity of implementation: on the server side separate trees are maintained for separate data stream, while on the client side the user need only store keys for the data he is consuming. This scheme, however, makes inefficient use of keys and does not scale well when there are many data streams or more complicated access relationship.

There are limited works that have addressed the dynamic group membership issues for certain special access scenarios. In [58], tree based traditional key management scheme is modified to fit the Bell-LaPedulla [59] confidentiality model. These schemes, however, cannot be generalized to solve more complicated hierarchical access control problems.

In this chapter, we present a *multi-group key management* scheme that maintains the keying material for all members with different access privileges using an integrated key graph. In particular, the hierarchical group access control problem is formulated in Section 3.2. The centralized multi-group key management is presented in Section 3.3, 3.4, and 3.5. Particularly, Section 3.3 describes the construction of the integrated key graph and the rekey algorithm. Section 3.4 analyzes the performance of the proposed scheme and the asymptotical behavior. Section 3.5 provides the simulation results and compares the proposed scheme with existing tree-based solutions in various application scenarios. Finally, the contributory key management scheme that uses the integrated key graph is presented in Section 3.6.

3.2 Hierarchical Access Control for Group Communications

3.2.1 System description

Let $R = \{r_1, r_2, \cdots\}$ denote the set of *resources* in the system. In the group communication scenario, the data of each resource are transmitted in one multicast session, which is associated with a multicast address and a multicast routing tree [2]. Thus, each resource corresponds to one multicast data stream. From data transmission points of view, a *Data Group* (DG) is defined as a set of users who have access to a resource. Thus, the users belonging to the same multicast session form a DG. It is clear that the DGs may have overlapped membership because users may subscribe multiple resources. From access control points of view, a *Service Groups* (SG) is defined as a set of users who can access the exactly same set of resources. SGs do not have overlapped membership. In this chapter, the DGs are denoted by $\{D_1, D_2, \cdots, D_M\}$, where M is the total number resources, and the SGs are denoted by $\{S_1, S_2, \cdots, S_I\}$, where I is the total number SGs. It is easy to prove that $I \leq 2^M - 1$.

The access relationship between the resources and the SGs is usually described by *capability lists*, which record the set of resources that can be accessed by each SG. Here are two examples illustrating typical access relationship in group communication.

Example 1. Multimedia applications distributing data in multi-layer format [38].

- *Resources*: {base layer (r_1), enhancement layer 1 (r_2), enhancement layer 2 (r_3)}.

- *Service Groups*: {users subscribing basic quality (S_1), users subscribing moderate quality (S_2), users subscribing high quality (S_3)}.
- *Capability lists*: S_1 access $\{r_1\}$; S_2 access $\{r_1, r_2\}$; S_3 access $\{r_1, r_2, r_3\}$.

Example 2. Multicast programs containing several related services.

- *Resources*: {News (r_1), Stock quote (r_2), Traffic/Weather (r_3)}.
- *Service Groups*: Users can subscribe any combination of the resources. Thus, there are total 7 SGs, denoted by $S_1, S_2, \cdots, S_7$.
- *Capability lists*: S_1 access $\{r_1\}$; S_2 access $\{r_2\}$; S_3 access $\{r_3\}$; S_4 access $\{r_1, r_2\}$; S_5 access $\{r_1, r_3\}$; S_6 access $\{r_2, r_3\}$; S_7 access $\{r_1, r_2, r_3\}$.

Besides capability list, *access matrix* is also used to describe the access relationship. In particular, the element on the i^{th} row and m^{th} column of the access matrix, denoted by $a_{i,m}$, is

$$a_{i,m} = \begin{cases} 1, & \text{if the SG } S_i \text{ can access the resource } r_m \\ 0, & \text{otherwise} \end{cases},$$

where $i = 1, \cdots, I$ and $m = 1, \cdots, M$.

Based on those definitions, the group size of SGs and DGs must satisfy:

$$n(D_m) = \sum_{i=1}^{I} a_{i,m} \cdot n(S_i), \tag{3.1}$$

where $n(S_i)$ is the number of users in SG S_i and $n(D_m)$ is the number of users in DG D_m.

3.2.2 Security requirements

Group communication often involves dynamic membership. In the applications containing multiple multicast sessions, users not only join or leave service, as addressed in the single multicast session scenario, but also may switch between the SGs by subscribing or dropping data streams. It is noted that the users' join/departure/switching behaviors are associated with the changes in their personal access privilege but do not affect the access relationship between the SGs and the resources.

We introduce the notation $S_i \rightarrow S_j$ that represents a user switching from the SG S_i to the SG S_j. To simplify future notations, S_0 is defined as a virtual service group containing users who cannot access any resources. Thus, $S_0 \rightarrow S_i$ represents a user joining the SG S_i, and $S_i \rightarrow S_0$ represents a user leaving the group communication from the SG S_i.

Similar as the single session access control problem addressed by traditional key management schemes [3], the hierarchical group access control should guarantee the following security requirements.

- The users in the SG S_i have and only have access to the resources $\{r_m, \forall\ m : a_{i,m} = 1\}$.
- When a user $S_i \rightarrow S_j$,
  - This user cannot access the future content of the resources $\{r_m, \forall\ m : a_{i,m} = 1 \text{ and } a_{j,m} = 0\}$. This property can be referred to as the forward secrecy [31].
  - This user cannot access the previous content of the resources $\{r_m, \forall\ m : a_{i,m} = 0 \text{ and } a_{j,m} = 1\}$. This property can be referred to as the backward secrecy [31].

3.2.3 Data encryption and hierarchical key management

In hierarchical access control scenario, there are two ways to encrypt and distribute multicast data. In the first method, resources are encrypted using separate keys, which are called *Data Group Keys*. The data group key used to encrypt resource r_m, denoted by K_m^D, is shared among the users in DG D_m. In this case, each resource is distributed in a single multicast session, and the users may subscribe to one or several multicast sessions according to their access privilege. The task of key management is to securely update and distribute $\{K_m^D, \ \forall m : a_{i,m} = 1\}$ to the users in S_i, where $i = 1, 2, \cdots, I$.

In the second method, the users in each SG share a secrete key called the *Service Group Key* and the multicast sessions are formed based on SGs. In particular, the users in S_i share the service group key K_i^S and form one multicast session. In this multicast session, the resources $\{r_m, \forall m : a_{i,m} = 1\}$ are encrypted by K_i^S and transmitted to the users in S_i. In this case, one resource may be distributed in several multicast sessions while being encrypted by different service group keys. The task of key management is to securely distribute and update K_i^S for the users in SG S_i. Compared with the first method, this method obviously consumes more bandwidth for data transmission. On the other hand, since users subscribe to only one multicast session, the task of key management for the second method can be solved by applying traditional key management for each SG separately.

In this work, the first encryption method is adopted because of its bandwidth efficiency. In order to guarantee forward and backward secrecy, when a user switches from SG S_i to S_j, it is necessary to

- update $\{K_m^D, \forall \ m : a_{i,m} = 0 \text{ and } a_{j,m} = 1\}$, such that this user cannot access the previous communication in corresponding DGs;

- and update $\{K_m^D, \forall\ m : a_{i,m} = 1 \text{ and } a_{j,m} = 0\}$, such that this user cannot access the future communication in corresponding DGs.

The focus of this work is to solve this hierarchical group key management problem efficiently.

3.3 Centralized Multi-group Key Management Scheme

Hierarchical group access control can be achieved in either centralized or contributory manner. While the contributory solution will be discussed in Section 3.6, this section and the following two sections will be dedicated to the centralized schemes.

3.3.1 Employing independent key trees to achieve hierarchical access control

To reduce the communication, computation and storage overhead, tree structure is widely used in centralized key management schemes to maintain the keying material and coordinate the key generation [4, 16–21] (see Section 4.2.3 and 1.3.2).

When using tree-based schemes to achieve hierarchical group access control, a separate key tree must be constructed for each DG, with the root being the data group key and the leaves being the users in this DG. This approach is referred to as the *Independent-tree* key management scheme, and is illustrated in Figure 3.1. This scheme does not exploit the relationship among the subscribers and makes inefficient use of keys because of the overlapped DG membership. As an extreme example, if a user who subscribes all data streams leaves the service, key updating has to take place on all key trees.

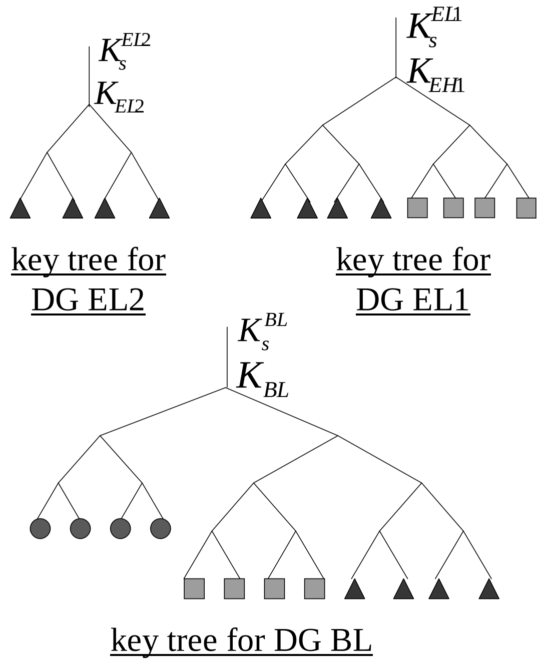


Figure 3.1: Independent-tree key management scheme for layered coded multimedia service

3.3.2 Multi-group key management scheme

To achieve hierarchical group access control, we propose a *multi-group* key management scheme that employs one integrated key graph accommodating key materials of all users. This key graph consists of several subtrees, and is constructed in three steps.

Step1: For each SG S_i, construct a subtree having the leaf nodes as the private keys of users in S_i and the root node as the service group key K_i^S. These subtrees are referred to as the *SG-subtrees.*

Step2: For each DG D_m, construct a subtree whose root is the DG key K_m^D and whose leaves are $\{K_i^S, \forall i : a_{i,m} = 1\}$. These subtrees are referred to as the *DG-subtrees.*

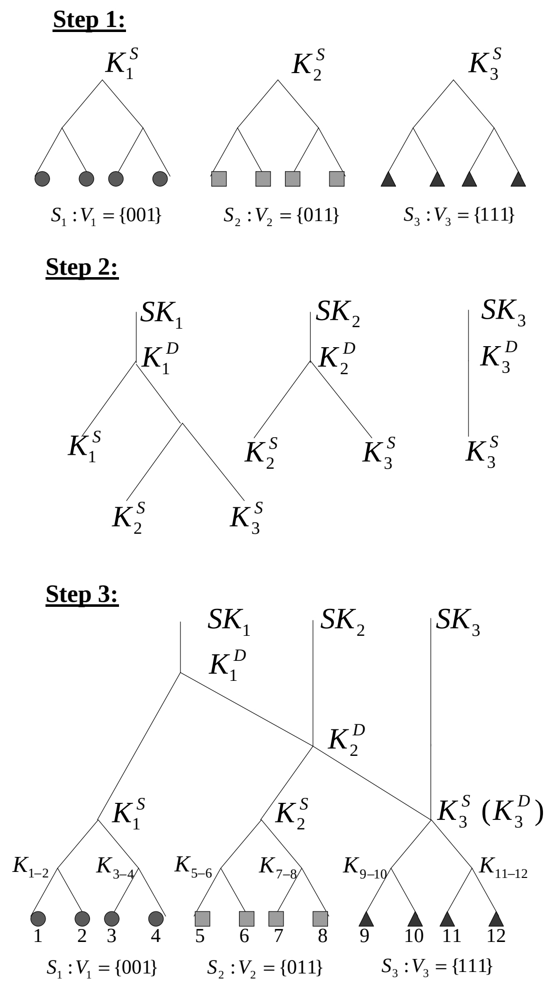


Figure 3.2: Multi-group key management graph construction

Step3: Generate the key graph by connecting the leaves of the DG-subtrees and roots of SG-subtrees.

This 3-step procedure is illustrated in Figure 3.2 for the services containing 3 layers and having 4 users in each SG. Some duplicated structures may appear on DG-subtrees, and they can be merged to reduce the number of keys on the key graph. In the example shown in Figure 3.2, K_3^S and K_3^D, which are on the same line, are merged. The DG-subtrees of D_2 and D_1 have the same structure that connect K_2^S and K_3^S. Thus, the parent node of K_2^S and K_3^S on DG-subtree of D_2 is merged with K_2^D.

This multi-group key graph can also be interpreted as M overlapped key trees, each of which has K_m^D as the root and the users in DG D_m as the leaves. Obviously, these M key trees can be used in the independent-tree scheme. This reveals the fact that the multi-group key graph removes the "redundancy" presented in the independent-tree scheme. Therefore, it can reduce the overhead associated with key updating.

As defined in [17], *keyset* refers to the set of keys associated with a edge node on the key graph and possessed by the user located at this edge node. In our key graph, the keyset of a user in SG S_i is the keys on the pathes from himself to the roots of the DG-subtrees of D_m for $\{m : a_{i,m} = 1\}$. It is noted that the keyset of users in S_0 is just an empty set.

Besides user join and departure, the rekey algorithm in the multi-group key management scheme must address users' relocation on the key graph. We describe the rekey algorithm for $S_i \rightarrow S_j$, which includes the cases for user join, departure, and switching SGs. First, the switching user is moved from the SG-subtree of S_i to a new location on the SG-subtree of S_j. Let ϕ_i denote the keyset associated with

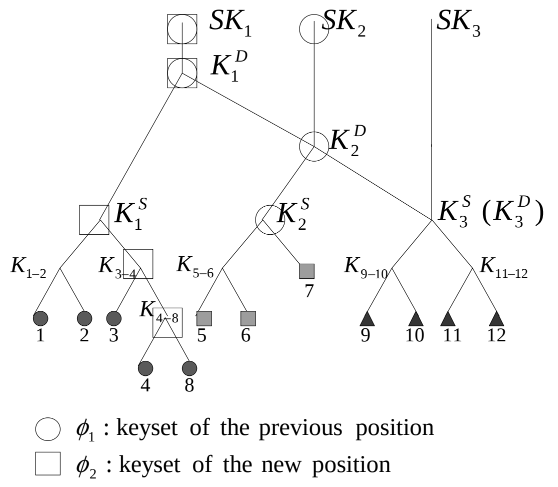


Figure 3.3: User relocation on the key graph

the user's previous position, and ϕ_j denote the keyset associated with the user's new position. Then,

- the KDC updates the keys in $\overline{\phi_i} \cap \phi_j$ using one-way functions, similar as the procedure for user join in [18],
- and, the KDC generates new versions of the keys in $\phi_i \cap \overline{\phi_j}$ and distributes these new keys encrypted by their children node keys from bottom to up, similar as the procedure for user departure in [18].

We illustrate this rekey algorithm through an example. Let user 8 switches from SG S_2 to S_1 (see Figure 3.3). On the SG-subtree of S_1, the leaf node associated with user 4 is split to accommodate user 8. Then, user 4 and 8 will share a new KEK, denoted by K_{4-8}. On the SG-subtree of S_2, user 7 will be moved up and occupy the node that is previously associated with K_{7-8}. In this case, ϕ_2 is $\{K_{7-8}, K_2^S, K_2^D, SK_2, K_1^D, SK_1\}$ and ϕ_1 is $\{K_{4-8}, K_{3-4}, K_1^S, K_1^D, SK_1\}$.

Let the notation x^{new} represent the new version of key x, $\{y\}_x$ represent the key y encrypted by key x, and u_k represent the private key of user k. Each key is associated with a ID, a version number and a revision number.

In this example, the KDC generates the new keys, K_{3-4}^{new} and $K_1^{S,new}$, from the old keys using a one-way function, and increases the revision numbers of those new keys. Thus, the user 1,2,3,4 will know about the key change when the data packet indicating the increase of the revision numbers first arrives, and compute the new keys using the same one-way function. No rekeying messages are necessary for distributing K_{3-4}^{new} and $K_1^{S,new}$.

Then, the KDC generates new keys, K_{4-8}^{new}, $K_2^{S,new}$, $K_2^{D,new}$, and SK_2^{new}, and distributes them through a set of rekeying messages as:

$$\{K_{4-8}^{new}\}_{u_8},\ \{K_{4-8}^{new}\}_{u_4},\ \{K_2^{S,new}\}_{K_{5-6}},\ \{K_2^{S,new}\}_{u_7}$$

$$\{K_2^{D,new}\}_{K_2^{S,new}},\ \{K_2^{D,new}\}_{K_3^S},\ \{SK_2^{new}\}_{K_2^{D,new}}$$

In this case, the rekeying message size is 7.

It is noted that $\overline{\phi_i} \cap \phi_j$ may contain the new KEKs that are created for accommodating the switching user. These new KEKs are encrypted by users' private keys and distributed through rekeying messages. In addition, $\phi_i \cap \overline{\phi_j}$ may contain KEKs that do not exist any more after the relocation of the switching user. Obviously, these keys are discarded.

3.4 Performance Measures and Analysis

Communication, computation and storage overhead associated with key updating are major performance criteria for key management schemes [3, 4, 17]. In the hierarchical access control scenario, we define the performance measures as:

- *Storage overhead at the KDC*: denoted by R_{KDC} and defined as the expected number of keys stored at the KDC.

- *Rekey overhead at the KDC*: denoted by M_{KDC} and defined as the expected amount of rekeying messages transmitted by the KDC.

- *Storage overhead of users*: denoted by $R_{u \in S_i}$ and defined as the expected number of keys stored by the users in SG S_i.

- *Rekey overhead of users*: denoted by $M_{u \in S_i}$ and defined as the expected amount of rekeying messages received by the users in SG S_i.

Here, R_{KDC} and $R_{u \in S_i}$ describe the storage overhead, while M_{KDC} and $M_{u \in S_i}$ reflect the usage of communication and computation resources.

3.4.1 Storage overhead

We first consider the storage overhead of a single key tree. Similar to most key management schemes [3, 4, 16–18], the key tree investigated in this work is fully loaded and maintained as balanced as possible by putting the joining users on the shortest branches.

Let $f_d(n)$ denote the length of the branches and $r_d(n)$ denote the total number of keys on the key tree when the key tree has degree d and accommodates n users. Since the key tree is balanced, $f_d(n)$ is either L_0 or $L_0 + 1$, where $L_0 = \lfloor \log_d n \rfloor$. Particularly,

- the number of users who are on the branches with length L_0 is $d^{L_0} - \lceil \frac{n - d^{L_0}}{d-1} \rceil$,

- and, the number of users who are on the branches with length $L_0 + 1$ is $n - d^{L_0} + \lceil \frac{n - d^{L_0}}{d-1} \rceil$.

Thus, the total number of keys on this key tree is calculated as:

$$r_d(n) \quad = \quad n + 1 + \frac{d^{L_0} - 1}{d - 1} + \lceil \frac{n - d^{L_0}}{d - 1} \rceil. \tag{3.2}$$

Using the fact that $\frac{n-d^{L_0}}{d-1} \le \lceil \frac{n-d^{L_0}}{d-1} \rceil < \frac{n-d^{L_0}}{d-1} + 1$, we have

$$\frac{dE[n]-1}{d-1} + 1 \le E[r_d(n)] < \frac{dE[n]-1}{d-1} + 2, \tag{3.3}$$

where the expectation, $E[.]$, is taken over the distribution of $n(D_m)$ and the length of the branches on the key trees. The left-hand-side equality is achieved when $\log_d(n)$ is an integer. In addition, since $\log_d(n)$ is a concave function and $\lfloor \log_d n \rfloor \le \log_d n$, it is clear that

$$E[f_d(n)] \le E[\log_d n] + 1 \le \log_d E[n] + 1. \tag{3.4}$$

With (3.3) and (3.4), we are ready to analyze the storage overhead. When using the separate key trees (i.e. independent-tree scheme), the KDC stores all keys on total M key trees, and users in S_i store subsets of keys on the key trees that are associated with D_m, for $\{m : t_m^0 = 1\}$. Thus,

$$R_{KDC}^{ind} = \sum_{m=1}^{M} E\left[r_d(n(D_m))\right], \tag{3.5}$$

$$R_{u\in S_i}^{ind} = \sum_{m=1}^{M} a_{i,m}\left(E[f_d(n(D_m))] + 1\right). \tag{3.6}$$

In the multi-group key management scheme, the DG-subtree of D_m has $c_m = \sum_i a_{i,m}$ leaf nodes. Before removing the redundancy on DG-subtrees, there are in total $\sum_{m=1}^{M} r_d(c_m)$ keys on DG-subtrees. Also, the total number of keys on the SG-subtrees is $\sum_{i=1}^{I} r_d(n(S_i))$. Merging duplicated structures of DG-subtrees can future reduce the number of keys on the key graph. Therefore, the storage overhead at the KDC is

$$R_{KDC}^{mg} \le \sum_{i=1}^{I} E[r_d(n(S_i))] + \sum_{m=1}^{M} E\left[r_d(c_m)\right]. \tag{3.7}$$

A user in SG S_i stores $f_d(n(S_i))$ keys on the SG-subtree and up to $\sum_{m=1}^{M} a_{i,m}(f_d(c_m)+1)$ keys on the DG-subtrees. Therefore, the users' storage overhead of the multi-group scheme is:

$$R^{mg}_{u \in S_i} \quad \leq \quad E[f_d(n(S_i))] + \sum_{m=1}^{M} a_{i,m}(E[f_d(c_m)] + 1). \tag{3.8}$$

We will demonstrate the storage overhead of the independent-tree and the multi-group key management in the applications containing multiple layers, as described in Example 1 in Section 3.2.1. In this case, $a_{i,m} = 1$ for $m \leq i$ and $a_{i,m} = 0$ for $m > i$. We also assume that each layer contains the same amount of users, denoted by $n(S_i) = n_0$. Thus, $n(D_m) = (M - m + 1)n_0$. Using (3.6) and (3.8), the users' storage overhead is calculated as:

$$R^{ind}_{u \in S_i} = \sum_{m=1}^{i} \left(E[f_d((M - m + 1) \cdot n_0)] + 1\right), \tag{3.9}$$

$$R^{mg}_{u \in S_i} \leq E[f_d(n_0)] + \sum_{m=1}^{i} \left(E[f_d(M - m + 1)] + 1\right). \tag{3.10}$$

When the group size is large, i.e. $n_0 \to \infty$, (3.4)(3.9) and (3.10) lead to

$$R^{ind}_{u \in S_i} \sim O(i \cdot \log(n_0)), \;\; R^{mg}_{u \in S_i} \sim O(\log(n_0)). \tag{3.11}$$

Using (3.5) and (3.7), the storage overhead at the KDC is calculated as:

$$R^{ind}_{KDC} \quad = \quad \sum_{m=1}^{M} E[r_d(m \cdot n_0)], \tag{3.12}$$

$$R^{mg}_{KDC} \quad \leq \quad M \cdot E[r_d(n_0)] + \sum_{m=1}^{M} E[r_d(m)]. \tag{3.13}$$

From (3.3), it is seen that $\lim_{n \to \infty} r_d(n) = \frac{d}{d-1} n$. Therefore,

$$R^{ind}_{KDC} \quad \sim \quad O(\frac{d}{d-1} \frac{M(M+1)}{2} n_0), \tag{3.14}$$

$$R^{mg}_{KDC} \quad \sim \quad O(\frac{d}{d-1} M \cdot n_0). \tag{3.15}$$

By using the integrated key graph instead of the separate key trees, the multi-group key management scheme reduces the storage overhead of both the KDC and the users. As indicated in (3.14) and (3.15), the storage advantage of the proposed scheme becomes larger when the system contains more SGs, i.e. requiring more levels of access control. The proposed scheme in fact scales better when the number of layers (M) increases.

3.4.2 Rekey overhead

In this section, we calculate the amount of rekeying messages transmitted by the KDC when one user switches from S_i to S_j, denoted by $C_{i,j}$. It is noted that the rekey overhead, M_{KDC} and $M_{u \in S_i}$, can be calculated from $C_{i,j}$, as long as the users' statistical joining/leaving/switching model is given.

Switching from S_i to S_j is equivalent to adding the subscription to the DG $\{D_m, \forall m : a_{i,m} = 0 \text{ and } a_{j,m} = 1\}$ and dropping the subscription to the DG $\{D_m, \forall m : a_{i,m} = 1 \text{ and } a_{j,m} = 0\}$. When using the tree-based key management schemes, the rekeying message size is calculated as:

$$C_{ij}^{ind} = \sum_{m=1}^{M} \max(a_{i,m} - a_{j,m}, 0) \cdot (d \cdot f_d(n(D_m))) \,. \tag{3.16}$$

We can see that the term $(\max(a_{i,m} - a_{j,m}, 0))$ equals to 1 only when $a_{i,m} = 1$ and $a_{j,m} = 0$. When this term equals to 1, $d \cdot f_d(n(D_m)$ rekeying messages are necessary to update keys on the key tree associated with the DG D_m.

In the multi-group key management scheme, when a user switches from S_i to S_j and $i \neq j$,

- The amount of messages that update the keys on the SG-subtree of S_i is up to $(d \cdot f_d(n(S_i)) - 1)$.
- The amount of messages that convey the KEK created for accommodating the switching/join user on the SG-subtree of S_j is always less than 2.

- If this user drops the subscription of the DG D_m, i.e. $(\max(a_{i,m} - a_{j,m}, 0)) = 1$, the amount of rekeying messages that update keys on the DG-subtree of D_m is up to $(d \cdot f_d(c_m) + 1)$.
- If this user remains the subscription of the DG D_m, i.e. $a_{i,m} = a_{j,m} = 1$, we need up to $(d \cdot f_d(c_m))$ rekeying messages to update keys on the DG-subtree of D_m.

Therefore, when using the multi-group scheme and $i \neq j$, we have

$$\begin{aligned} C_{ij}^{mg} \leq \textstyle\sum_{m=1}^{M} (&\max(a_{i,m} - a_{j,m}, 0) \cdot (d \cdot f_d(c_m) + 1) \\ &+ a_{i,m} a_{j,m} d \cdot f_d(c_m)) + d \cdot f_d(n(S_i)) + 1, \end{aligned} \tag{3.17}$$

Similar as in Section 3.4.1, we analyze the rekey overhead in a multi-layer scenario with $n(S_i) = n_0$. In this case, the rekeying message size for one user departure, i.e. $S_j \to S_0$, is computed from (3.16) and (3.17) as:

$$\begin{aligned} C_{0j}^{ind} &= \sum_{m=1}^{j} d \cdot E[f_d((M - m + 1) n_0)], \qquad (3.18) \\ C_{0j}^{mg} &\leq d \cdot E[f_d(n_0)] + 1 + \sum_{m=1}^{j} (d \cdot E[f_d(M - m + 1)] + 1) \,. \end{aligned}$$

When $n_0 \to \infty$, we can see that

$$C_{0j}^{ind} \sim O(i \cdot d \cdot \log(n_0)), \; C_{0j}^{mg} \sim O(d \cdot \log(n_0)). \tag{3.19}$$

The comprehensive comparison between the proposed scheme and the independent-tree scheme will be presented in Section 3.5.

3.5 Simulations and Performance Comparison

In this section, the proposed multi-group key management scheme is compared with the existing tree-based key management schemes in various application scenarios.

3.5.1 Statistical dynamic membership model

In [50] [51], it has been shown that the users' arrival process and membership duration of MBone multicast sessions can be modelled by Poisson and exponential distribution respectively, in a short period of time. In this work, we use this Poisson arrival and exponential distribution duration model, and assume that when a user switches between SGs, the SG that he switches to depends only on his current SG.

Therefore, the users' statistical behavior can be described by an embedded Markov chain [53]. Particularly, there are a total of $I + 1$ states, denoted by $\widetilde{S_i},\ i = 0, \cdots, I$. When a user is in the SG S_i, he is in the state $\widetilde{S_i}$. After a user enters state $\widetilde{S}_i$, i.e. subscribes or switches to SG S_i, this user stays at state $\widetilde{S}_i$ for time T_i, which is governed by an exponential random variable. When time is up, the user moves to state $\widetilde{S}_j$. The selection of $\widetilde{S}_j$ only depends on the current state $\widetilde{S}_i$ and is not related to previous states.

In practice, it is usually not necessary to update keys immediately after membership changes. Many applications allow the join/departure users receive limited previous/future communications [39]. For example, in video streaming applications, a joining user may receive a complete group-of-picture (GOP) [38] although partial of this GOP already been transmitted before his subscription. Those situations prefer *batch rekeying* [39] that postpones key updating such that the rekeying overhead is reduced by adding or removing several users altogether.

In this work, batch rekeying is implemented as updating keys periodically. The time between key updates is fixed and denoted by B_t. For the users who join/leave/switch SGs in the time interval $((k-1)B_t, kB_t]$, the key updating will take place at time kB_t, where k is a positive integer. From the key updating points of view, with batch rekeying, we can prove that the previous continuous Markov

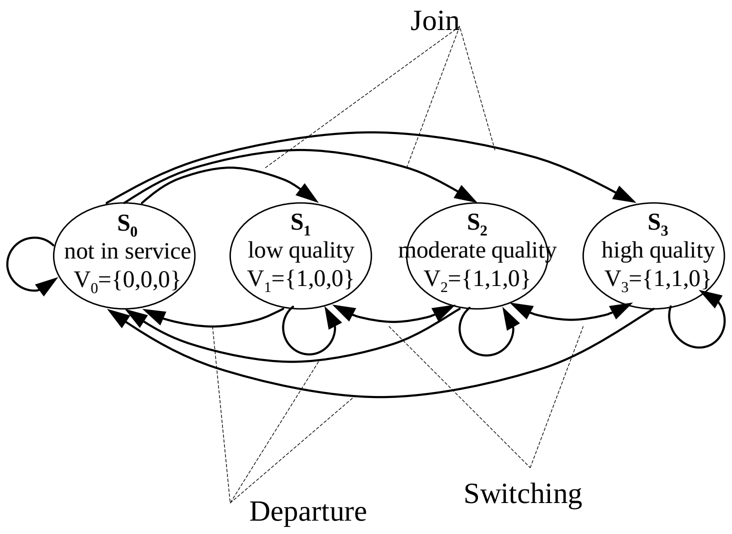


Figure 3.4: Discrete Markov chain model for multi-layer applications.

model can be simplified as a discrete Markov chain model [53], as illustrated in Figure 3.4. In this model,

- The transition matrix is denoted by $P = [p_{ij}]_{(I+1)\times(I+1)}$, where p_{ij} is the probability that one user moves from SG S_i to S_j in the time interval $(kB_t, (k+1)B_t]$ given that this user is in S_i at time kB_t.

- The n-step transition probability matrix is denoted by $P(n)$, and obviously, $P(n) = P^N$. The element at the i^{th} row and j^{th} column of $P(n)$ is denoted by $p_{ij}(n)$.

- The stationary state probability is a 1-by-$(I + 1)$ vector, denoted by $\pi = [\pi_0, \pi_1, \cdots, \pi_I]$.

In practice, most group applications have the following properties.

- $p(n)_{0j} \neq 0$ for some positive finite n and for any j because users should be able to subscribe to any SGs.

- $p(n)_{i0} \neq 0$ for some positive finite n and for any i because users should be able to leave from any SGs.

- $p_{ii} > 0$ because users can always stay in his current SG.

- The mean recurrence time [53] of the state $\widetilde{S}_0$ is finite because the expected time that a user stays in the group communication is finite.

Because of these properties, this Markov chain is irreducible, aperiodic and positive recurrent. As a result, the stationary state probability mass function (pmf) exists [53] and is the unique solution of

$$\pi P = \pi, \text{ and } \sum_i \pi_i = 1 \ . \tag{3.20}$$

3.5.2 Performance with different group size

We first study the applications containing multiple layers (see Example 1 in Section3.2.1) where users in SG S_i can access DG $D_1, D_2, \cdots, D_i$. In the simulation, the transition matrix is chosen as follows.

- Users join different SGs with the same probability, i.e. $P_{0j} = \alpha, \forall j > 0$.

- Users leave different SGs with the same probability, i.e. $P_{i0} = \beta, \forall i > 0$.

- While a user is in the service, he adds/drops only one DG at a time, i.e. $P_{i,j} = 0, \forall i, j > 0$ and $|i - j| > 1$. Also, users switch between SGs with the same probability, i.e. $P_{i,j} = \gamma, \forall i, j > 0$ and $|i - j| = 1$.

Thus, the transition matrix is described by only three variables. For example, the multi-layer service with $M = 3$ has the transition matrix as:

$$P = \begin{bmatrix} 1-3\alpha & \alpha & \alpha & \alpha \\ \beta & 1-\beta-\gamma & \gamma & 0 \\ \beta & \gamma & 1-\beta-2\gamma & \gamma \\ \beta & 0 & \gamma & 1-\beta-\gamma \end{bmatrix}$$

In all simulations, batch rekeying is applied and the key trees are binary. The initial state is chosen as the stationary state, i.e. S_i contains $N_0\pi_i$ users at the beginning of the service.

In Figure 3.5, 3.6, 3.7 and 3.8, the multi-group scheme and the independent-tree scheme are compared for varying group size, N_0. The results are averaged over 300 realizations, and the number of layers is 4. In those simulations, we chose $\alpha = 0.005$, $\beta = 0.01$, and $\gamma = 0.001$.

Figure 3.5 shows that the storage overhead at the KDC, R_{KDC}, increases linearly with the group size. This result can be verified by (3.3)(3.5) and (3.7). In the case when $M = 4$, the multi-group scheme reduces R_{KDC} by more than 50%.

Figure 3.6 shows that the users' storage overhead, $R_{u\in S_i}$, increases linearly with the logarithm of the group size. This can be verified by (3.9) and (3.10). The users who subscribe only one layer have the similar storage overhead in both schemes. For the uses who subscribe multiple layers, the multi-group scheme results in less storage overhead than the independent-tree scheme.

The KDC's rekeying overhead, R_{KDC} and the users' rekey overhead, $R_{u\in S_i}$ are shown in Figure 3.7 and 3.8, respectively. In both cases, the multi-group scheme reduces the rekey overhead by more than 50%.

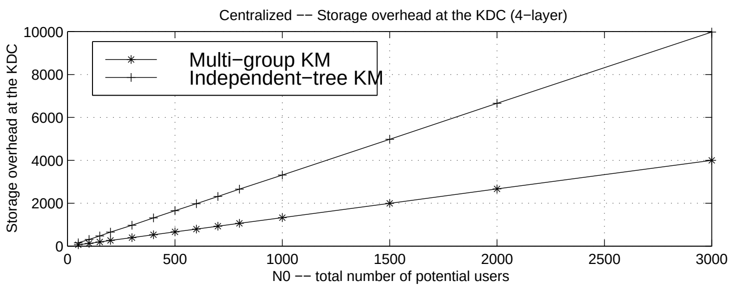


Figure 3.5: Storage overhead at the KDC

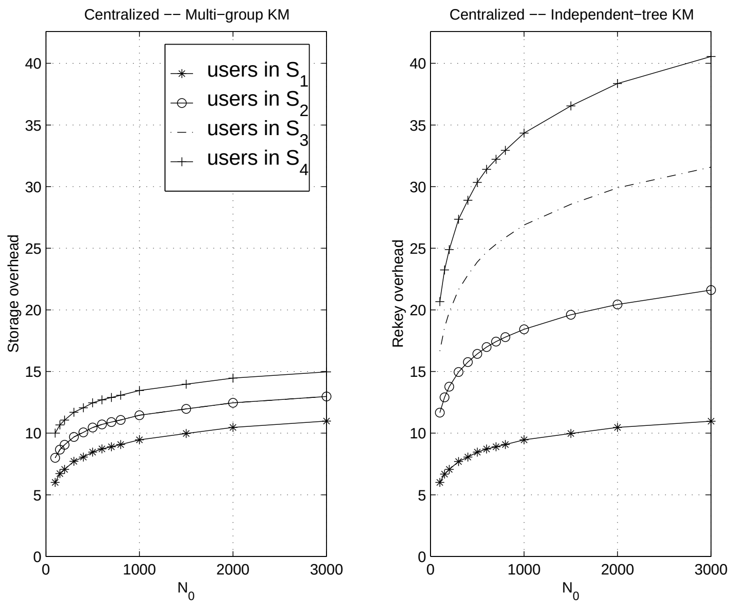


Figure 3.6: Storage overhead at the users in each SG

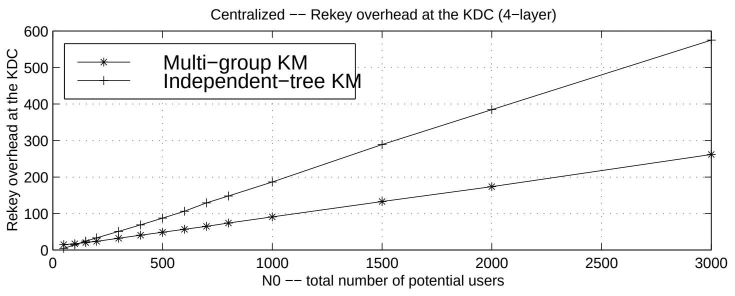


Figure 3.7: Rekey overhead at the KDC

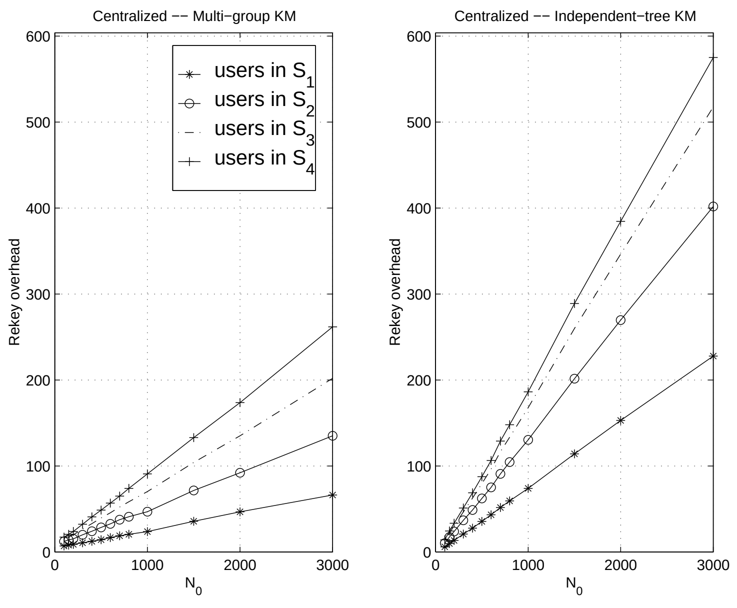


Figure 3.8: Rekey overhead at the users in each SG

3.5.3 Scalability

Next, we change the number of layers (M) while maintaining roughly the same number of users in the service by choosing the join probability α as $0.02/M$. The values of β and γ are the same as those in Section 3.5.2.

Figure 3.9(a) and Figure 3.10(a) show the storage and rekey overhead at the KDC, respectively. When M increases, the storage and rekey overhead of the multi-group scheme do not change much, while the overhead of the independent-tree scheme increases linearly with M. It is clear that the multi-group scheme scales better when M increase. By removing the redundancy in DG membership, the scale of the key graph mainly depends on the group size, not the number of layers or services. On the other hand, by constructing M separate key trees, the independent-tree scheme requires larger storage and rekey overhead when M increases even when N_0 is fixed.

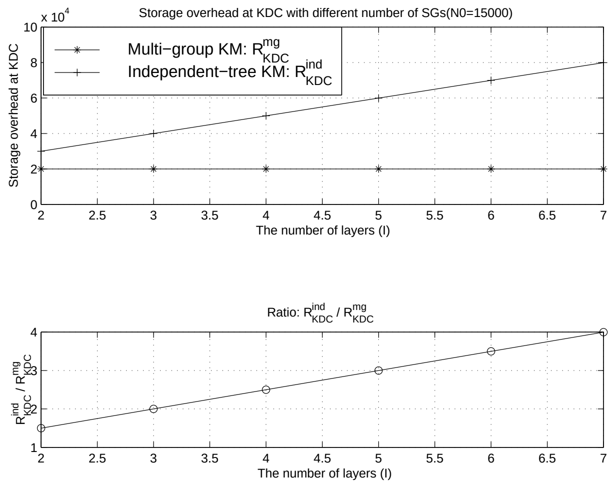


Figure 3.9: Storage overhead at the KDC with different number of SGs

Figure 3.9(b) shows that the ratio between R^{ind}_{KDC} and R^{mg}_{KDC} increases linearly with M, which agrees with (3.14) and (3.15). Similarly, the ratio between M^{ind}_{KDC} and M^{mg}_{KDC} increases linearly with M, as shown in Figure 3.10(b).

3.5.4 Performance with different transition probability

In the previous simulations, we set $\gamma = 0.1\beta$, which means that the users are more likely to leave the service than to switch SGs. Figure 3.11 shows the rekey overhead with different values of γ. Remember that γ describes the probability of user switching between SGs. In this simulation, $M = 4$, $N_0 = 1000$, and the values of α and β are the same as those in the previous experiments.

When γ is very small, the multi-group scheme reduces the rekey overhead by about 50%, as we have shown in the previous simulations. When γ is less than 2β, the advantage of the multi-group scheme decreases with the increase of γ. This is because the multi-group scheme introduces larger rekey overhead when users switch SGs by simply subscribing more DGs. To see this, let a user move from

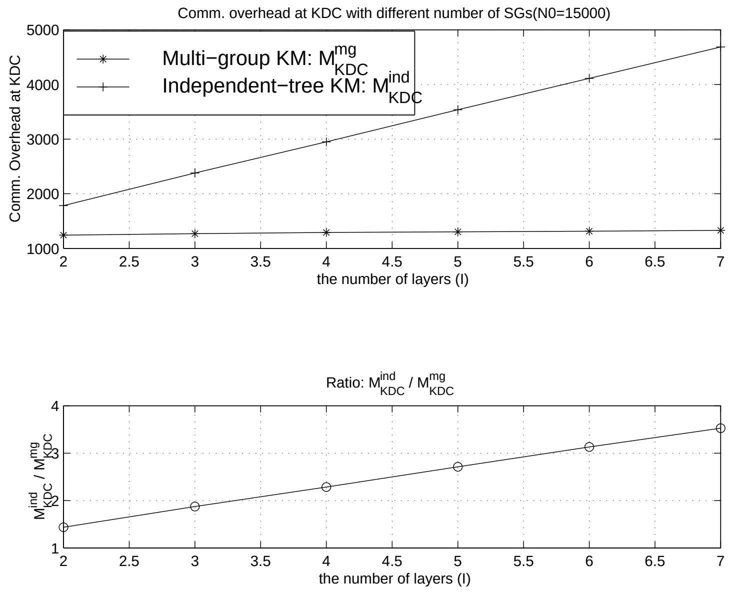


Figure 3.10: Rekey overhead at the KDC with different number of SGs

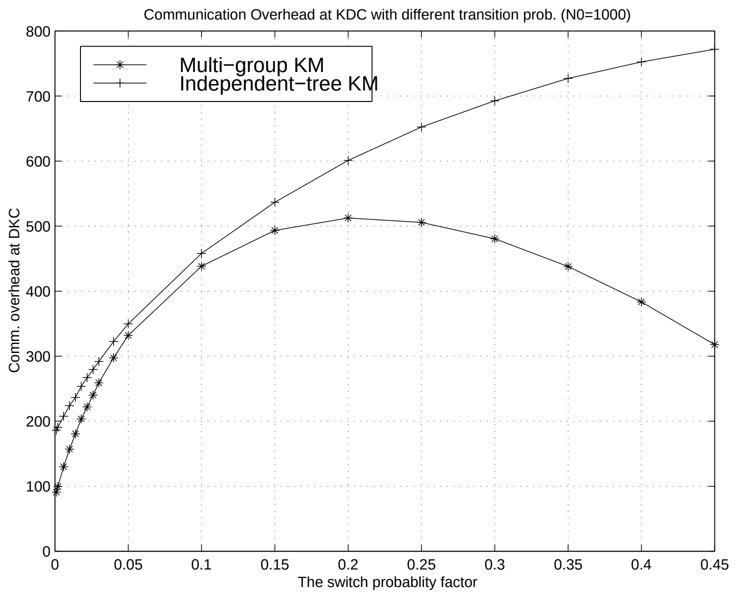


Figure 3.11: Rekey overhead at the KDC with different transition probability

SG S_1 to SG S_2. When using the independent-tree scheme, this user only needs to be added to the key tree associated with the DG D_2 and no rekeying messages are necessary. When using the multi-group scheme, we need to update keys on the SG-subtree of S_1 and the DG-subtree of D_1. Therefore, the performance gain reduces when more users tend to switch SGs.

When γ continues to increase, however, the rekey overhead of the multi-group scheme decreases. Particularly, when $\gamma = 0.45$, which describes the scenario where users are much more likely to switch SGs than to stay in the current SG or leave the service, the performance gain of the multi-group scheme is about 50% again. This phenomena is due to the fact that the size of the SG-subtree is greatly reduced when a significant potion of users are switching away from this SG. In this case, removing a large potion of users from the key tree using batch rekeying requires less rekeying messages than just removing several users.

3.5.5 Simulation of multi-service applications

We also simulated the multi-service scenario illustrated in Example 2 (Section 3.2.1), which contains 3 DGs and 7 SGs. The users can subscribe any combination of DGs and switch to any SGs. Here, the transition matrix is 8 by 8, with $P_{j0} = 0.01, \forall j > 0$ and $P_{i,j} = 0.00017, \forall i, j > 0$ and $i \neq j$. N_0 is fixed to be 1500. The values of $P_{0i}, \forall i > 0$, are adjusted such that the SGs contain varying number of users while $(\sum_{i=1}^{I} P_{0i})$ is maintained to be the same.

The horizontal axis in Figure 3.12 is the ratio between the number of users subscribing more than one DGs and the number of users subscribing only one DG. Larger is this ratio, more overlap is in DG membership. Figure 3.12 shows that the advantages of the multi-group scheme is larger when more users subscribe multiple

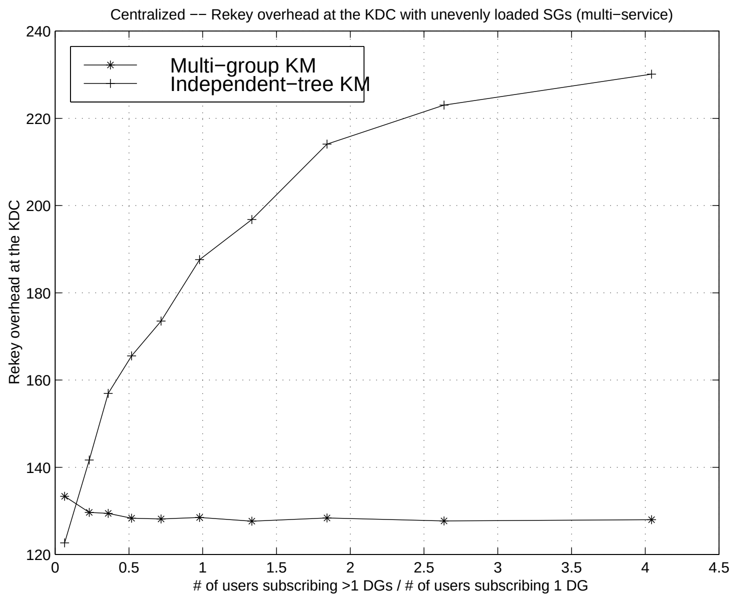


Figure 3.12: Rekey overhead at the KDC with unevenly loaded SGs in multi-service applications

DGs.

3.6 Contributory Multi-group Key Management

The multi-group key management schemes can be extended to the contributory environment by using the same graph construction procedure presented in Section 3.3.2. Similar as in the centralized environments, separate key trees for each DG must be constructed when using existing tree-based contributory schemes [31–33], and the multi-group contributory schemes maintains one integrated key graph for all users.

The key establishment protocols are straightforward extensions from the existing protocols in tree-based contributory schemes [31–33]. When users join/leave/switch, the set of keys that need to be recalculated is the same as that need to be updated

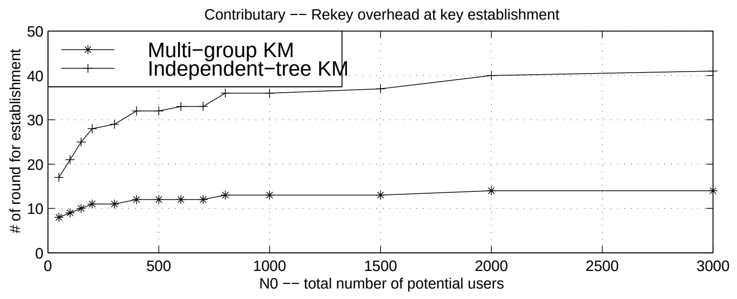


Figure 3.13: The total number of rounds performed to establish the group key

in the protocols presented in Section 3.3.2. The new keys are recalculated by applying the DH protocol between the users who are under the left child node and the users who are under the right child node from bottom to up.

For contributory key management schemes, the number of rounds is usually used to measure the communication, computation, and latency [60] associated with key establishment and updating [30–32].

With the same simulation setup as that in Section 3.5.2, the performance of the independent-tree and multi-group contributory key management schemes are compared for varying group size. Figure 3.13 shows the total number of rounds to establish the group key, which reflects the latency in key establishment [60]. Figure 3.14 shows the number of rounds performed by the users in each SG, which describes the users' computation overhead. In each round, a user performs two modular exponentiations. With the same simulation setup as that in Section 3.5.2, Figure 3.15 shows the number of rounds for key updating for with different number of layers. Compared with the tree-based contributory schemes, the multi-group contributory scheme significantly reduces the computation and latency associated with key establishment and updating. The advantage of the multi-group contributory scheme is larger when M increases.

In this chapter, we designed a multi-group key management scheme that achieves

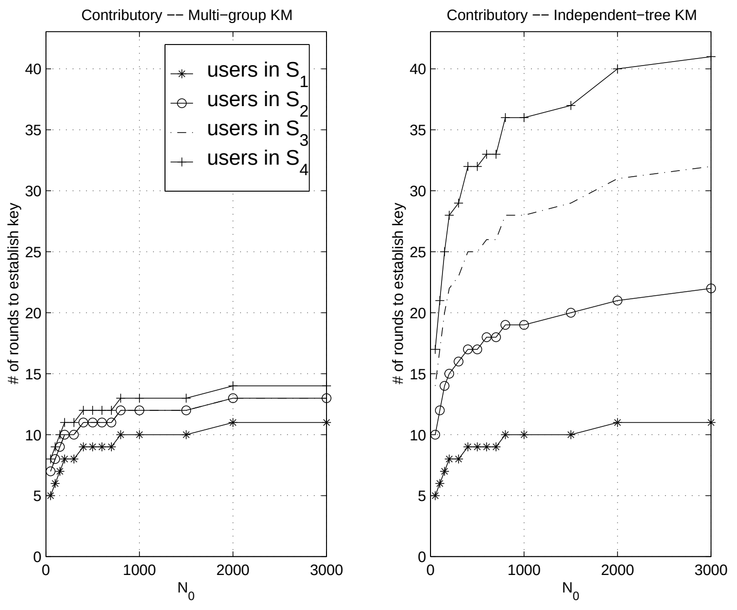


Figure 3.14: The number of rounds performed by the users in each SG for key establishment

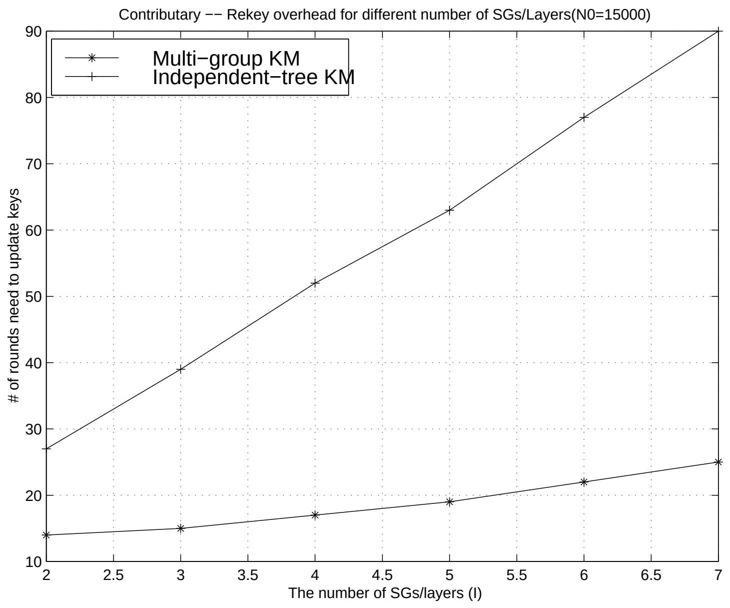


Figure 3.15: The number of rounds performed to establish the group key with different number of SGs/layers

hierarchical group access control in secure group communications, where multiple data streams are distributed to group members with various access privileges. The proposed multi-group key management scheme employs an integrated key graph to maintain keying material, and uses a generalized rekey algorithm that allow users subscribing/dropping the group communications as well as changing access levels. Compared with traditional tree-based key management, the proposed scheme can greatly reduce the communication, computation, and storage overhead associated key establishment and update. Further, when the system contains more data streams, i.e. more complicated access relationship, the multi-group key management scheme achieves better scalability than tree-based schemes.

Chapter 4

Protecting Dynamic Group Information in Secure Multicast

4.1 Introduction

In order to design better security protocols, it is necessary to look at the security system from an adversarial point-of-view. By aggressively inspecting existing key management protocols, we discovered a weakness that has been overlooked during the design phase. That is, key management can disclose group dynamic information (GDI) to both insiders and outsiders. In this work, the group dynamic information particularly refers to a set of functions as:

- $N(t)$: the number of users in the multicast group at time t.
- $J(t_0, t_1)$: the number of users who join the service between time t_0 and t_1.
- $L(t_0, t_1)$: the number of users who leave the service between time t_0 and t_1.

In many group communications, GDI is confidential and should not be disclosed to either valid group members or outsiders. Such group applications widely exist in commercial as well as military applications, as discussed in Section 1.4. However, to acquire GDI by launching attacks on the key management schemes can be very simple as we will demonstrate. Instead of trying to break the encryption or compromise the key distribution center, the adversaries can subscribe to the service as regular users. In this case, they are referred to as the *inside attackers*. As we will show later in this chapter, inside attackers can obtain very accurate estimation of GDI by monitoring the messages conveying new key updating information, i.e. rekeying messages. Even if the adversaries cannot become valid group members, they still have the opportunities to steal GDI as *outside attackers* as long as they can observe the traffic and distinguish the rekeying messages and other data.

In this chapter, we demonstrate that the key management schemes can reveal the GDI easily and propose a framework of protecting GDI by introducing anti-attack technologies to key management. In particular, the attack strategies and the anti-attack method for the centralized schemes are presented in Section 4.2 and Section 4.3 respectively. In Section 4.4, the performance criteria of the proposed anti-attack method are derived and the optimization problem is formulated. Simulation results based on the user log data from real MBone sessions are shown in Section 4.5. The investigation on contributory key management schemes is presented in Section 4.6.

4.2 GDI Attacks on Centralized Key management

In centralized key management schemes there usually exists a centralized server, such as the service provider, who is trustful, well protected, and has the computation and storage ability to generate and distribute the decryption keys [3]. In this section, we investigate the attack strategies that aim to attack the centralized key management schemes for obtaining the dynamic group information. Two attack strategies for tree-based key management will be presented, followed by a discussion of vulnerability of other prevalent centralized key management schemes.

We consider a popular tree-based centralized key management scheme proposed in [18], whose rekeying process has been shown in Section 4.2.3. In brief, when a user leaves the group, all the keys on the path from this user to the root of the key tree are updated by conveying a set of rekeying messages, that have the basic format as one key encrypted by another key. When a user joins the group, all existing users compute the new key using a one-way function upon noticing the increased revision numbers of keys. No additional rekeying messages are necessary.

The rekeying procedure although has some differences, most tree-based centralized key management schemes [3, 4, 16–19] share two common properties. First, group members can distinguish the key updating process due to user join and that due to user departure. Second, rekeying message size is closely related with the group size. Due to these properties, the attackers can estimate $J(t_0, t_1)$ and $L(t_0, t_1)$ by examining the rekey processes, and estimate $N(t)$ directly from the rekeying messages size. Next, we illustrate these two types of attacks on the tree-based key management scheme presented in [18].

4.2.1 Attack A1: Estimation of the number of join/departure users by inside attackers

An inside attacker, like other regular users, processes K_s, K_ϵ, and a set of KEKs. He receives rekeying messages, decrypts the messages that are encrypted by his keys, and observes the rekeying message size without having to understand the content of all messages. Since the key updating process for user join and the process for user departure are different, he can estimate $J(t_0, t_1)$ and $L(t_0, t_1)$ using the following strategy:

- When receiving the rekeying message containing K_ϵ^{new} encrypted by one of his KEKs, he assumes that one user leaves the service.
- When observing the increase of the revision number of K_ϵ, he assumes that one user joins the service.

This strategy is effective when most users do not join/leave simultaneously and the keys are updated immediately once each user join/departure. Otherwise, more complicated techniques involving examining the rekeying message size shall be used. When this attack is successful, $N(t)$ can be calculated from $J(t_0, t_1)$ and $L(t_0, t_1)$ as:

$$N(t_1) = N(t_0) + J(t_0, t_1) - L(t_0, t_1). \tag{4.1}$$

Even if the attacker do not know the initial value of the group size, he obtains the changing trend of the group size.

4.2.2 Attack AII: Estimation of group size from rekeying message size

Besides using (4.1), the group size $N(t)$ can also be estimated directly from the rekeying message size. We will derive a Maximum Likelihood estimator for the attackers and then demonstrate the effectiveness of this estimator through simulations.

We assume that $N(t)$ does not change much within a short period of time. In this time period, there are W departure users who do not leave simultaneously.

Thus, the attacker makes W observations of the rekeying message size due to single user departure, denoted by $Msg = \{m_1, m_2, \cdots, m_w\}$.

Similar to most key management schemes [3,4,16–18], the key tree investigated in this work is fully loaded and maintained as balanced as possible by putting the joining users on the shortest branches. In the worst-case scenario, the attacker knows this property and the degree of the key tree, denoted by d. Then, the attacker can calculate the depth of the branch where the i^{th} leaving user was located before departure, denoted by L_i. Without losing information, the observed Msg is converted to $\{L_1 = l_1, L_2 = l_2, \cdots, L_W = l_W\}$, where $l_i = \lceil \frac{m_i+1}{d} \rceil$. Then, the Maximum Likelihood (ML) estimator is formulated as:

$$N_{ML} = \arg\max_n \; Prob\{L_1 = l_1, L_2 = l_2, \cdots, L_W = l_W | N(t) = n\}. \tag{4.2}$$

To solve (4.2), we introduce a set of new variables: $\{S_k\}_{k=L_{min}, L_{min}+1, \cdots, L_{max}}$, where S_k is the number of users who are on the branches with length k, L_{max} is the length of the longest branches, and L_{min} is the length of the shortest branches. It is obvious that

$$\sum_k S_k = n. \tag{4.3}$$

In addition, the length of the branches of a key tree must satisfy the Kraft inequality [54], i.e. $\sum_j d^{L_{max}-b_j} \leq d^{L_{max}}$, where b_j is the length of the branch on which the user j stays and $j = 1, 2, \cdots, n$. Thus, S_k, which equals to the number of elements in set $\{b_j : b_j = k\}$, must satisfy

$$\sum_k S_k d^{L_{max}-k} \leq d^{L_{max}}, \tag{4.4}$$

It can be verified that the equality is achieved when all intermediate nodes on the key tree have d children nodes. When the key tree is balanced and fully loaded, it is reasonable to approximate (4.4) by

$$\sum_k S_k d^{L_{max}-k} = d^{L_{max}}. \tag{4.5}$$

We assume that the leaving users are uniformly distributed on the key tree, and the number of users in the system is much larger than the number of leaving users, i.e. $N(t) >> W$. Then, the probability mass function (pmf) of L_i is

$$Prob\{L_i = k\ |n,\ S_k\} = \frac{S_k}{n}, \quad k = L_{min}, L_{min}+1, \cdots, L_{max}.$$

We assume that $L_i, i = 1, \cdots, W$ are i.i.d. random variables. Thus, the probability in (4.2) is calculated as:

$$Prob\{L_1 = l_1, L_2 = l_2, \cdots, L_W = l_W\ |N(t) = n,\ S_k\} = \prod_k \left(\frac{S_k}{n}\right)^{h(k)}, \tag{4.6}$$

where $h(k)$ denotes the number of elements in set $\{l_i : l_i = k\}$ and obviously, $\sum_k h(k) = W$. Then, the values of n and $\{S_k\}$ that maximize (4.6) under the constraint (4.3) and (4.5) are obtained using Lagrange multiplier as:

$$\{S_k\}_{ML} = \frac{n}{W} h(k), \tag{4.7}$$

$$N_{ML} = \frac{W}{\sum_k h(k) d^{-k}}. \tag{4.8}$$

This ML estimator was applied to simulated multicast services. As suggested in [50] [51], the user arrival process is modelled as poisson process, and the service duration is modelled as an exponential random variable. In Figure 4.1(a), 4.1(b), and 4.1(c), the estimated group size is obtained by using the estimator in (4.8), and compared with the true values of $N(t)$. These three plots are for different simulation settings. The entire service period is divided into four sessions. The model parameters, i.e. user arrival rate and average service time, are fixed within each session and vary in different sessions. In the i^{th} session, described by interval

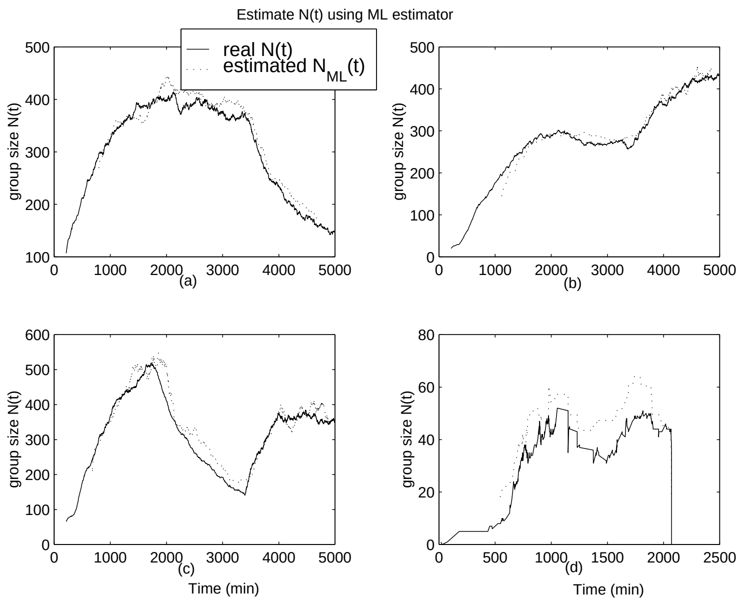


Figure 4.1: Performance of the ML estimator

$[t_{i-1}, t_i)$, the user arrival rate is λ_i and the average service time is μ_i. In all three cases, $[t_0, t_1, t_2, t_3, t_4]$ is chosen to be $[0, 200, 1600, 3200, 5000]$ minutes, and the initial group size is 0. In plot (a), $[\lambda_1, \lambda_2, \lambda_3, \lambda_4] = [0.5, 0.5, 0.5, 0.3]\text{min}^{-1}$ and $[\mu_1, \mu_2, \mu_3, \mu_4] = [1400, 800, 600, 400]\text{min}$. In plot (b), $[\lambda_1, \lambda_2, \lambda_3, \lambda_4]$ is chosen as $[0.1, 0.3, 0.2, 0.5]\text{min}^{-1}$, and $[\mu_1, \mu_2, \mu_3, \mu_4]$ is chosen as $[1500, 1500, 1000, 800]\text{min}$. In plot (c), $[\lambda_1, \lambda_2, \lambda_3, \lambda_4]$ is $[0.3, 0.7, 0.1, 0.9]\text{min}^{-1}$ and $[\mu_1, \mu_2, \mu_3, \mu_4]$ is chosen as $[1400, 800, 600, 400]\text{min}$. In addition, Figure 4.1(d) demonstrates the performance of the ML estimator, when it was applied to a real MBone audio session, CBC Newsworld on-line test, starting on Oct. 29. 1996 and lasted for about 5 days [61].

In all four cases, the changing trend of the group size is well captured by the attacker. It is also observed that the estimated group size tends to be larger than the true $N(t)$, which is due to the approximation that we replace (4.5) by (4.4). Although not perfect, this estimator is effective in helping the attackers to achieve many of their goals, such as analyzing audience behavior and monitoring the group size changes.

The inside attackers can launch both attack AI and AII. They obtain $J(t_0, t_1)$ and $L(t_0, t_1)$ using AI, and the initial value $N(t_0)$ using AII. Then, $N(t)$ can be obtained by using either (4.1) or (4.8), or jointly.

It has been shown that the rekeying messages must be delivered reliably and in a timely manner in order to guarantee the quality of service [62]. Therefore, it is possible that rekeying messages are treated differently from the regular data in terms of error control, or even transmitted in a reliable multicast channel separated from the channel used for transmitting multicast content. This provides an opportunity for outsiders to separate the rekeying messages and the multicast content. Thus, the outsiders may also launch attack AII directly by monitoring the transmission of the rekeying messages.

It should be noted that the performance of the attack AI and AII degrades when many users join/leave simultaneously. It will be shown in Section 4.3 that the rekeying message size still reveals a significant amount of information on GDI even when multiple users are removed from or added to the key tree together.

4.2.3 Vulnerability of popular centralized key management schemes

The attack methods described in Section 4.2.1 and 4.2.2 can be tailored to many other key management schemes. When the inside attacker can separate the rekeying messages for user join and those for user departure, they launch *AI type attacks*. When the amount of rekeying messages is largely depends on the group size, attackers can launch *AII type attacks*, although the estimator may be slightly different from (4.8). In this section, we review several key management schemes and discuss their vulnerability to AI and AII type attacks.

Since protecting GDI is not part of the design goal in traditional key management schemes, it is not surprising that some schemes reveal GDI in a very direct way. For example, in the approach proposed in [22], a security lock is implemented based on the Chinese remainder theorem and the length of the lock is proportional to the number of users. Thus, $N(t)$ is obtained by measure the length of the lock, which is the simplest AII type attack.

Tree-based key management schemes have been known for their efficiency in terms of the usage of communication, computation and storage resources. Many tree-based schemes, such as [4, 17–19], are similar to that described in Section . In these cases, both AI and AII type attacks can be applied. In [16, 20, 21], another class of tree-based schemes were presented to further reduce the communication overhead by introducing the dependency among keys, such as using one-way function trees. In these schemes, only AII type attacks are suitable.

Besides the tree-based scheme described in Section , VersaKey framework [18] also includes a centralized flat scheme. When a user joins or leaves the group, the rekeying message size equals to the length of the binary representation of the user ID, which can be independent of $N(t)$. Thus, this key management scheme is resistant to both AI and AII type attacks. This scheme, however, is vulnerable to collusion attacks. That is, the KDC cannot update keys without leaking new key information to the leaving user, who has a collusion partner in the group. Although the GDI is protected, this scheme cannot protect the multicast content well when collusion attacks are likely.

In Iolus [23], a large group is decomposed into a number of subgroups, and the trusted local security agents perform admission control and key updating for the subgroups. This architecture reduces the number of users affected by key updating due to membership changes. Since the key updating is localized within each subgroup, the attacker can only obtain the dynamic membership information of the subgroup that he belongs to.

The idea of clustering was introduced in [24] to achieve the efficiency by localizing the key updating. The group members are organized into a hierarchical clustering structure. The cluster leaders are selected from group members and perform partial key management. Since the cluster leaders establish keys for the cluster members through pair-wise key exchange [24], the cluster members cannot obtain GDI of their clusters. However, the cluster leaders naturally obtain the dynamic membership information of their cluster and all clusters below by participating key management. In [24], the cluster size is chosen from 3 to 15. Therefore, this key management scheme can be applied only when a large potion of group members are trusted to perform key management and obtain GDI.

In Chapter 2, we have presented a topology-matching key management (TMKM) scheme that reduced the communication overhead associated with key updating by matching the key tree with the network topology and localizing the transmission of the rekeying messages. In this scheme, group members receive only the rekeying messages that are useful for themselves and their neighbors. Thus, they only obtains the local GDI by using AI or AII type attacks.

As a summary, Table 4.1 lists various key management schemes and their vulnerability to AI and AII type attacks. We can see that the AII type attacks are effective for stealing GDI or local GDI from many key management schemes. Two schemes, flat VersaKey [18] and the clustering [24], are resistant to these attacks. Their usage, however, are limited by the fact that they are either not resistant to

Centralized Key Management Schemes		Is attack AII Effective?	Is Attack AI Effective?
Tree Based	Key Graph [17], Wallner98 [4], Tree-based scheme in VersaKey framework [18] Embedding [19]	Yes	Yes
	One-way function tree [20] Improve Key Revocation [16] ELK [21]	Yes	No
Flat	Security lock [22]	Yes	–
	Flat centralized scheme in VersaKey framework [18]*	No	No
Local security agents	Iolus [23]	Local	Local
	Clustering [24]*	No	No
Others	TMKM [63]	Local	Local

Table 4.1: Vulnerability of popular centralized key management schemes

collusion attacks or must put trust in a large number of cluster leaders. Therefore, it is very important to investigate the anti-attack techniques to protect group dynamic information that are compatible with a variety of key management schemes.

4.3 Anti-attack Techniques

We have discussed two types of attacks that can steal GDI from centralized key management schemes. This discussion, however, does not cover all aspects of the key management schemes that can reveal group dynamic information. For example, the number of KEKs possessed by the inside attacker equals to the depth of the key tree and reveals at least the order of the group size. We can also show that the IDs of the keys reveal the structure of the key tree. Thus, new attack methods may emerge in the future. Therefore, we propose an anti-attack framework that is robust to various types of attacks and compatible with most centralized key management schemes.

We first introduce the concept of *Batch Rekeying* that plays an important role in our anti-attack technique. As proposed in [39], batch rekeying is to postpone the update of keys such that several users can be added to or removed from the key tree altogether. Compared with updating keys immediately after each user join or departure, batch rekeying reduces the communication overhead at the expense of allowing the joining/leaving user to access a small amount of information before/after his join/departure.

In this work, batch rekeying is implemented as periodic updating of keys and the time between key updates are fixed and denoted by B_t. Particularly, the users who join or leave the group in the time interval $[(k-1)B_t, kB_t]$, are added to or removed from the key tree together at time kB_t. Then, the notations of GDI functions are simplified as: $J(k) = J((k-1)B_t, kB_t)$, $L(k) = L((k-1)B_t, kB_t)$, and $N(k) = N(kB_t)$.

Since the AI type attacks are effective only when users are added to or removed from the key tree individually, utilizing batch rekeying can fight against the AI type attacks. However, batch rekeying alone is not enough to fight against the AII type attacks. Figure 4.2 shows some simulation results for the batch rekeying when B_t is set to be 5 minutes. Simulation setup is similar to that in Section 4.2.2. The solid

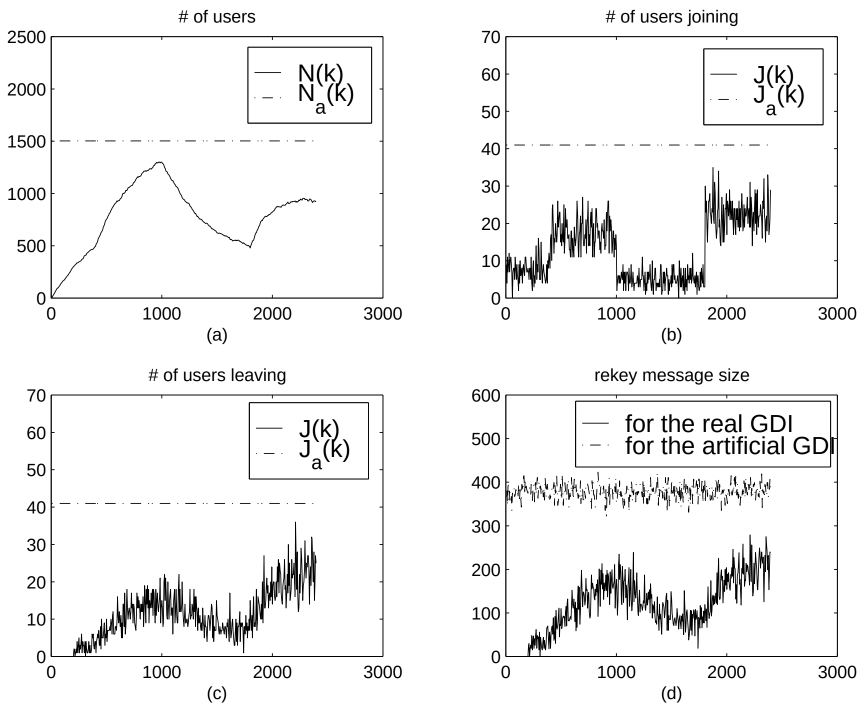


Figure 4.2: The anti-attack scheme using phantom users and batch rekeying

line in Figure 4.2(a), 4.2(b), 4.2(c), 4.2(d) represent the $N(k)$, $J(k)$, $L(k)$ and the rekeying message size, respectively. One can see that the rekeying message size is closely related to $L(k)$ and reflects the trend of $N(k)$. A large amount of information about $N(k)$ and $L(k)$ can be obtained by the attackers from examining the rekeying message size.

Besides using batch rekeying, we propose to insert phantom users into the system. These phantom users, as well as their join and departure behavior, are created by the KDC in such a way that the combined effects of the phantom users and the real users lead to a new rekeying process, called *observed rekeying process*, which is observed by the attackers. An important goal is for the system to produce an observed rekeying process that reveals the least amount of information about the GDI.

Let $N_a(k)$ denote the total number of the real and phantom users, and $J_a(k)$ and $L_a(k)$ denote the total number of the real and phantom users who join/leave the service respectively. $N_a(t)$, $J_a(k)$, and $L_a(k)$ are referred to as the *artificial GDI*. From the key management points of view, the phantom users are treated the same as the real users. They occupy leaf nodes on the key tree, and they are associated with a set of KEKs that are updated when they virtually join or leave the group. Thus, the observed rekeying process only depends on the artificial GDI.

We first consider choosing the artificial GDI as a set of constant functions, that is,

$$J_a(k) = L_0, \quad L_a(k) = L_0, \quad N_a(k) = N_0. \tag{4.9}$$

By doing so, the observed rekeying process does not leak the information about the changing trend of the real GDI. However, the perfect flat artificial GDI functions in (4.9) may not be achievable. Since the real GDI functions are random processes, it is possible that the predetermined L_0 and N_0 are not large enough such that the artificial GDI cannot be maintained as straight lines. For example, when $N(k) > N_0$, $N_a(k)$ cannot be the predetermined value N_0 because the number of phantom users must be non-negative. In fact, the artificial GDI functions must satisfies four requirements: (r1) $N_a(k) \geq N(k)$, (r2) $L_a(k) \geq L(k)$, (r3) $J_a(k) \geq J(k)$, and (r4) $N_a(k) = N_a(k-1) + J_a(k) - L_a(k)$. In this work, we choose the artificial GDI functions as:

$$\begin{aligned} N_a(k) &= \max\{N(k), N_0\} &(4.10)\\ J_a(k) &= \max\{J(k), L(k), L_0\} &(4.11)\\ L_a(k) &= N_a(k-1) - N_a(k) + J_a(k) &(4.12)\end{aligned}$$

When $N(k) \leq N_0$, $L(k) \leq L_0$, and $J(k) \leq L_0$, equation (4.10)-(4.12) are equivalent to (4.9). The artificial GDI functions in (4.10)-(4.12) obviously satisfy requirement (r1) (r3) and (r4). Next, we prove that the requirement (r2) is also satisfied.

- When $N(k) > N_0$, it follows that

  $$L_a(k) = N_a(k-1) - N_a(k) + J_a(k) \quad \geq \quad L(k) = N(k-1) - N(k) + J(k),$$

  using the fact that $N_a(k-1) \geq N(k-1)$, $N_a(k) = N(k)$, and $J_a(k) \geq J(k)$.

- When $N(k) \leq N_0$, one can see that

  $$L_a(k) \geq J_a(k) \geq L(k),$$

  using the fact that $N_a(k-1) \geq N_0$ and $J_a(k) \geq L(k)$.

It shall be noted that there are many other ways to choose the artificial GDI functions. The proposed anti-attack scheme supports any artificial GDI functions that satisfy the requirement (r1)-(r4).

Given the artificial GDI functions, the KDC creates phantom users and performs key management as follows.

(1) Determine N_0 and L_0 based on the system requirements and the users' statistical behavior. The criteria for selecting N_0 and L_0 will be presented in Section 4.4.

(2) Before the service starts, create N_0 phantom users and establish a key tree to accommodate them. Set index $k = 1$.

(3) While the service is not terminated, execute the following:

   - Record user join and departure requests in the time period $((k-1)B_t, kB_t]$, and obtain $J(k)$ and $L(k)$. During this time, the current session key is sent to the joining users such that they can start receiving the multicast content without delay.

– At time kB_t, the KDC creates $J_a(k) - J(k)$ phantom users joining the service, and then selects $L_a(k) - L(k)$ phantom users in the current system and makes them leave. Following the key updating procedure presented in any existing key management schemes, the KDC updates corresponding keys for real and phantom users' join and departure. The number of total real and phantom users are maintained to be $N_a(k)$.

– Set $k = k + 1$.

Figure 4.2(a), 4.2(b), and 4.2(c) illustrate the real GDI ($N(k)$, $L(k)$, $J(k)$) and the artificial GDI ($N_a(k)$, $L_a(k)$, $J_a(k)$) for a simulated multicast service. The simulation results of communication overhead, i.e. the rekeying message size, is shown in Figure 4.2(d), where the solid line represents the case without phantom users and the dash line represents case when the proposed anti-attack method is applied. We can see that the observed process reveals very limited information about the real GDI. Not surprisingly, the communication overhead increases, which is a disadvantage of utilizing phantom users.

Utilizing phantom users and batch rekeying is not the only solution to the problem of GDI leakage. There are other techniques that can protect GDI from one or several attacks. For example, embedding rekeying messages into the multicast content [19] can prevent outside attackers to launch the AII type attacks. Using the same rekeying procedure for user join and departure is also a good way to prevent the AI type attacks. In addition, the KDC can generate faked rekeying messages to prevent the AII type attacks, which is different from the proposed anti-attack scheme where the key tree reserves slots for the phantom users and all rekeying messages have meanings.

Compared with other techniques, using phantom users and batch rekeying has two major advantages. First, the proposed anti-attack scheme resists to a variety of attacks. Since the real GDI are concealed *before* the rekeying messages are generated, the attackers only see the artificial GDI from the observed rekeying process unless they break the encryption or compromise the KDC. Second, the proposed scheme does not rely on specific rekeying algorithms and is compatible with existing key management schemes.

4.4 Performance Measure and Optimization

In this section, we define two performance criteria and evaluate the performance of the proposed anti-attack technique. The criteria are (a) the amount of information leaked to the attackers measured by mutual information, and (b) the communication overhead introduced by the phantom users. We study the tradeoff between these two metrics and provide a framework of choosing proper amount of phantom users, described by the parameter L_0 and N_0 in (4.10)-(4.12).

4.4.1 The leakage of GDI

We use mutual information to measure the leakage of the GDI, which is independent of the attack strategies adopted by the attackers and represents the maximum amount of information that the attackers can possibly obtain. Let T be the total number of key updating, that is, the service duration is TB_t. Then, the real GDI is described by a set of random variables as

$$R = \{N(1), \cdots, N(T), J(1), \cdots, J(T), L(1), \cdots, L(T)\}, \qquad (4.13)$$

and the artificial GDI is

$$A = \{N_a(1), \cdots, N_a(T), J_a(1), \cdots, J_a(T), L_a(1), \cdots, L_a(T)\}. \tag{4.14}$$

The mutual information, $I(R; A)$, describes the reduction in the uncertainty of the real GDI (R) due to the knowledge of the artificial GDI (A) [54]. Therefore, the leakage of the GDI can be measured by

$$I(R; A) = H(A) - H(A|R), \tag{4.15}$$

where $H(.)$ and $H(.|.)$ denote the entropy and conditional entropy, respectively.

Equation (4.10) - (4.12) indicate that the artificial GDI is a set of deterministic functions of the real GDI. Thus, the conditional entropy in (4.15) equals to zero, i.e. $H(A|R) = 0$. Since $L_a(k)$ is directly computed from $J_a(k)$, $N_a(k)$ and $N_a(k-1)$ in (4.12), the terms $L_a(1), L_a(2), \cdots, L_a(T)$ can be removed from the expression of the entropy of A, i.e. $H(A) = H(N_a(1), \cdots, N_a(T), J_a(1), \cdots, J_a(T))$. Then, the upper bound of $I(R; A)$ is calculated as:

$$\begin{aligned} I(R; A) &= H(N_a(1), \cdots, N_a(T), J_a(1), \cdots, J_a(T)) \\ &\leq \sum_k H(N_a(k)) + \sum_k H(J_a(k)). \end{aligned} \tag{4.16}$$

The equality is achieved when $\{N_a(k), J_a(k), k = 1, \cdots, T\}$ are mutually independent. It is noted that the GDI at time kB_t and the GDI at time $(k+1)B_t$ can be approximately independent when B_t is large and the group is high dynamic. In these cases, (4.16) provides a tight upper bound of $I(R; A)$.

We introduce $p_{N_k}(n)$ and $p_{N_{ak}}(n)$ to denote the pmf of $N(k)$ and $N_a(k)$, respectively. From (4.10), one can see that

$$p_{N_{ak}}(n) = \begin{cases} \sum_{x=0}^{N_0} p_{N_k}(x), & n = N_0 \\ p_{N_k}(n), & n > N_0 \\ 0, & o.w. \end{cases}$$

Then,

$$H(N_a(k)) = -(1-\epsilon_N^k)\log(1-\epsilon_N^k) - \sum_{n=N_0+1}^{\infty} p_{N_k}(n)\log p_{N_k}(n), \tag{4.17}$$

where $\epsilon_N^k = 1 - \sum_{x=0}^{N_0} p_{N_k}(x)$. Similarly, let $p_{J_k}(x)$, $p_{J_{ak}}(j)$, and $p_{L_k}(y)$ denote the pmf of $J(k)$, $J_a(k)$, and $L(k)$, respectively. We then have,

$$H(J_a(k)) = -\sum_j p_{J_{ak}}(j)\log p_{J_{ak}}(j), \tag{4.18}$$

and,

$$p_{J_{ak}}(j) = \begin{cases} (1-\epsilon_J^k)(1-\epsilon_L^k), & j = L_0 \\ p_{J_k}(j)\sum_{y=0}^{j-1} p_{L_k}(y) + p_{L_k}(j)\sum_{x=0}^{j-1} p_{J_k}(x) + p_{J_k}(j)p_{L_k}(j), & j > L_0 \\ 0, & o.w. \end{cases} \tag{4.19}$$

where $\epsilon_J^k = 1 - \sum_{x=0}^{L_0} p_{J_k}(x)$ and $\epsilon_L^k = 1 - \sum_{y=0}^{L_0} p_{L_k}(y)$. Given the pmf of the real GDI functions, the upper bound of $I(R;A)$ is calculated from (4.16)-(4.19). Since the observed rekeying process is determined by the artificial GDI, the mutual information between the observed process and the real GDI is bounded by $I(R;A)$ due to the data processing theory [54]. Therefore, $I(R;A)$ is the upper bound of the amount of information that can be possibly obtained by the attackers.

From (4.10)-(4.12), one can see that the artificial GDI reveals the real GDI when $N(k) > N_0$, $L(k) > L_0$, or $J(k) > L_0$. We define *overflow probability* as the probability that the artificial GDI cannot be straight lines, i.e. $1 - \min_k(1-\epsilon_N^k)(1-\epsilon_L^k)(1-\epsilon_J^k)$. Besides the mutual information, overflow probability can be a more visualized complementary measure for the leakage of the GDI. When the overflow probability is zero, the calculation in (4.16)-(4.18) leads to the result that $I(R;A) = 0$, which indicates the prefect protection of the real GDI.

4.4.2 Communication overhead

Communication overhead, measured by the rekeying message size, is one of the major performance criteria of key management schemes [3] [4]. We introduce the notation $M(L, N, d)$ as the expected value of the rekeying message size when removing L users from the key tree that contains total N users and has degree d. We assume that the leaving users are uniformly distributed on a full loaded and balanced key tree. Then, there are d^l KEKs at the l^{th} level of the key tree for $l = 1, \cdots, D-2$ and $D = \lceil \log_d N \rceil$, and the number of the KEKs at the $(D-1)^{th}$ level is $s_1 = \lceil \frac{N-d^{D-1}}{d-1} \rceil$.

Let α^l be the number of KEKs need to be updated at level l when L user leaves the service. Then, $M(L, N, d)$ is expressed as:

$$M(L, N, d) \;=\; E\left[\sum_{l=0}^{D-1} \alpha_l\right] = \sum_{l=0}^{D-1} E[\alpha_l] \tag{4.20}$$

We introduce the notation $B(b, i, a)$, which is equivalent to the expected number of non-empty boxes when putting i items in b boxes with repetition where each box can have at most a items. The detailed calculation of $B(b, i, a)$ is provided in the Appendix. We can show that

$$E[\alpha_l] \;=\; d \cdot B(d^l, L, \frac{N}{d^l}), \quad 0 \le l \le D-2, \tag{4.21}$$

$$E[\alpha_{D-1}] \;=\; (d-1) \sum_{\widetilde{L}=1}^{L} \binom{s_1}{\widetilde{L}} \binom{N-s_1}{L-\widetilde{L}} \Big/ \binom{N}{L} B(s_1, \widetilde{L}, d) \tag{4.22}$$

Using the fact that $\lceil \frac{i}{a} \rceil \le B(b, i, a) \le \min(b, i)$ (see Appendix), we can derive the upper bound of the $M(L, N, d)$ as:

$$M(L, N, d) \le dL \log_d(N). \tag{4.23}$$

This upper bound indicates that the communication overhead increases linearly with the number of departure users and with the logarithm of the group size.

Let C_r and C_a be the average communication overhead for rekey process based on real GDI and the artificial GDI, respectively. Then, the extra communication overhead introduced by the proposed anti-attack technique is:

$$C_a - C_r = \frac{1}{T}\sum_{k=1}^{T} M(L_a(k), N_a(k), d) - \frac{1}{T}\sum_{k=1}^{T} M(L(k), N(k), d). \tag{4.24}$$

When the overflow probability is small, (4.24) can be approximated by:

$$C_a - C_r \approx M(L_0, N_0, d) - \frac{1}{T}\sum_{k=1}^{T} M(L(k), N(k), d). \tag{4.25}$$

4.4.3 System optimization

From the system design points of view, parameter L_0 and N_0 should be chosen such that the leakage of the GDI is minimized while the extra communication overhead do not exceed certain requirements. When the overflow probability is small, the optimization problem is formulated as:

$$\min_{N_0, L_0} \sum_k H(N_a(k)) + \sum_k H(J_a(k)) \tag{4.26}$$

subject to:

$$M(L_0, N_0, d) \le \beta, \tag{4.27}$$

where β is the maximum allowed communication overhead per key updating. We can show that $H(N_a(k))$ in (4.18) is monotonous non-increasing with N_0; $H(J_a(k))$ in (4.17) is monotonous non-increasing with L_0; and the communication overhead $M(L_0, N_0, d)$ in (4.20) is non-decreasing with L_0 and N_0. Therefore, the optimization problem is simplified as:

$$\min_{L_0} \left(\sum_k H(N_a(k)) + \sum_k H(J_a(k)) \right) \Big|_{N_0 = M^{-1}(\beta)|_{L_0, d}} , \tag{4.28}$$

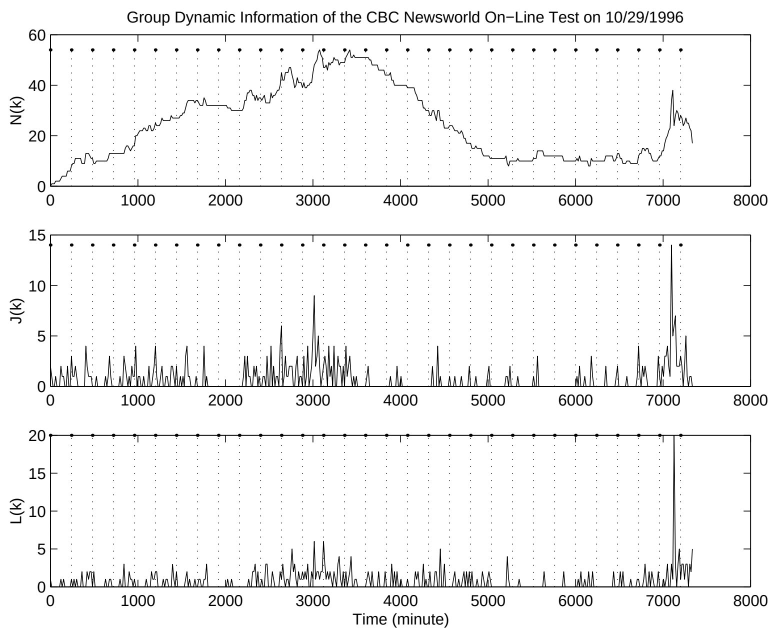


Figure 4.3: The GDI of a long audio session in MBone

where $M^{-1}(\beta)|_{L_0,d}$ is the largest value of N_0 that satieties (4.27) with given L_0 and d. Fortunately, the number of departure users between two key updates is usually not a large number in practice. Thus, the searching space for parameter L_0 is not large and this optimization problem can be solved by full search.

4.5 Simulations of the anti-attack scheme

Mlisten[1], a tool developed at Georgia Institute of Technology, can collect the join/leave time for the multicast group members in MBone [50] sessions. Using this tool, the characteristics of the membership dynamics of MBone multicast sessions has been studied in [50] [51].

The proposed anti-attack scheme is applied to the data collected in 1996 [61].

[1]available at www.cc.gatech.edu/computing/Telecomm.mbone

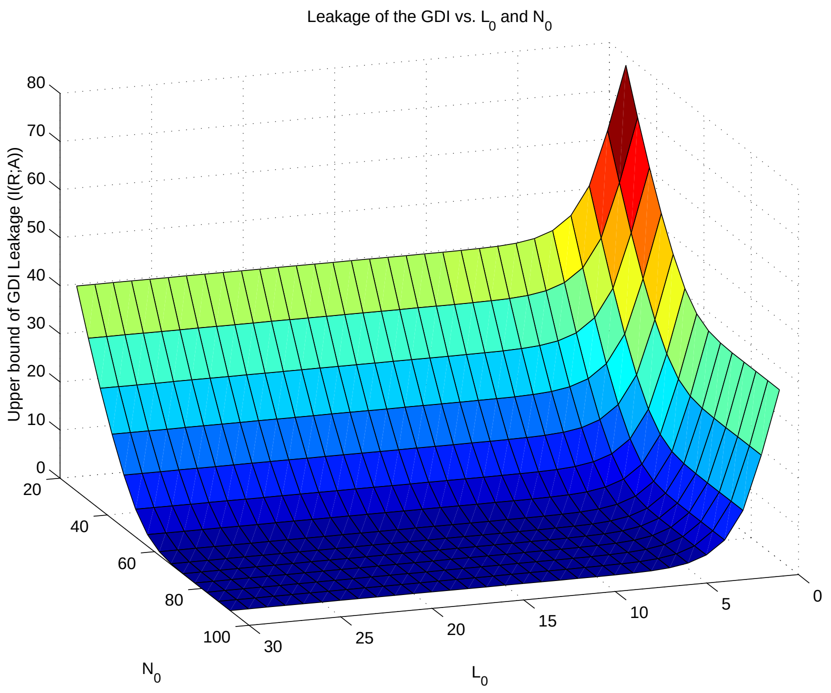


Figure 4.4: Upper bound of the GDI leakages

Particularly, we selected one audio session that started on Oct. 29th and lasted for about 5 days and 20 hours. Figure 4.3 shows the $N(k)$, $L(k)$ and $J(k)$ of this session, where the B_t is chosen to be 15 minutes.

It is suggested that the users statistical behavior, such as inter-arrival and membership durations, can be modelled by exponential distribution in a short period of time [50]. In the simulation, the entire service time is divided into non-overlapped sections, as illustrated in Figure 4.3. The length of these sessions is set to be 4 hours. To simplify the analysis, it is assumed that $N(k)$, $L(k)$ and $J(k)$ are stationary and ergodic Poisson processes in each session. Then, we can calculate the GDI leakage using (4.16)-(4.19).

Figure 4.4 and Figure 4.5 demonstrate the upper bound of mutual information (see (4.16)) and the communication overhead $M(L_0, N_0, d)$ for different values of L_0

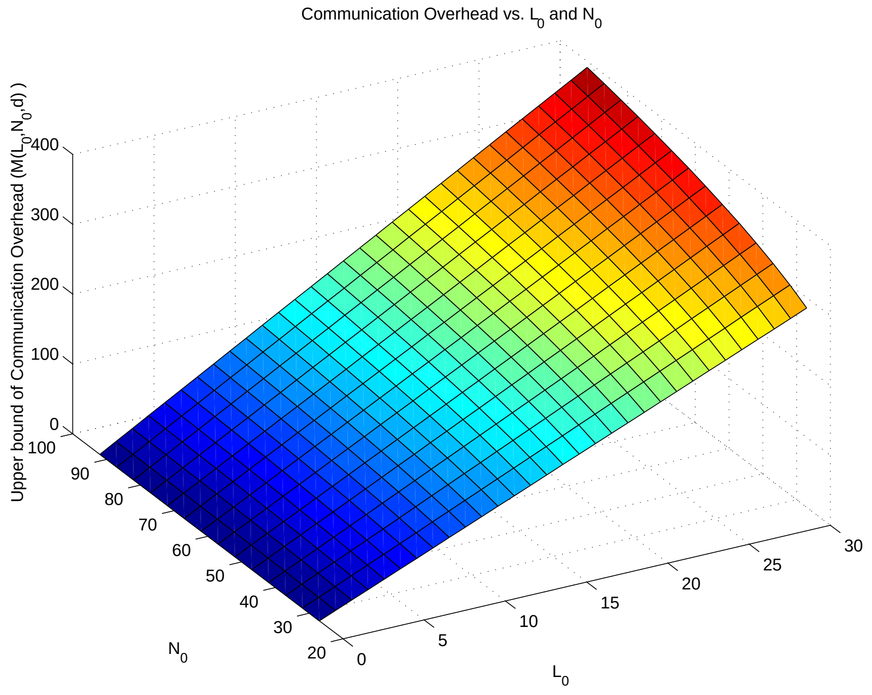


Figure 4.5: Communication overhead $M(L_0, N_0, d)$

and N_0, respectively. We can see that communication overhead is a non-decreasing function with L_0 and N_0, while the GDI leakage is a non-increasing function with L_0 and N_0. This verifies the argument in Section 4.4.

Figure 4.6 illustrates the solution of the optimization problem. Figure 4.6(a) shows the maximum value of N_0 that satisfies the communication overhead constraint in (4.27) with fixed L_0, i.e. $N_0 = max\{N : M(L_0, N, d) \leq \beta\}$, where β is chosen to be 50 in this example. As discussed in Section 4.4, the optimal values of L_0 and N_0 must be on this curve. Therefore, the upper bound of the GDI leakage, $\sum_k H(N_a(k)) + \sum_k H(J_a(k))$, is evaluated only at $(L_0, N_0 = max\{N : M(L_0, N, d) \leq \beta\})$, which is shown in Figure 4.6(b). The optimal values of L_0 and N_0 are also marked.

Figure 4.7 shows the tradeoff between the communication overhead and the GDI

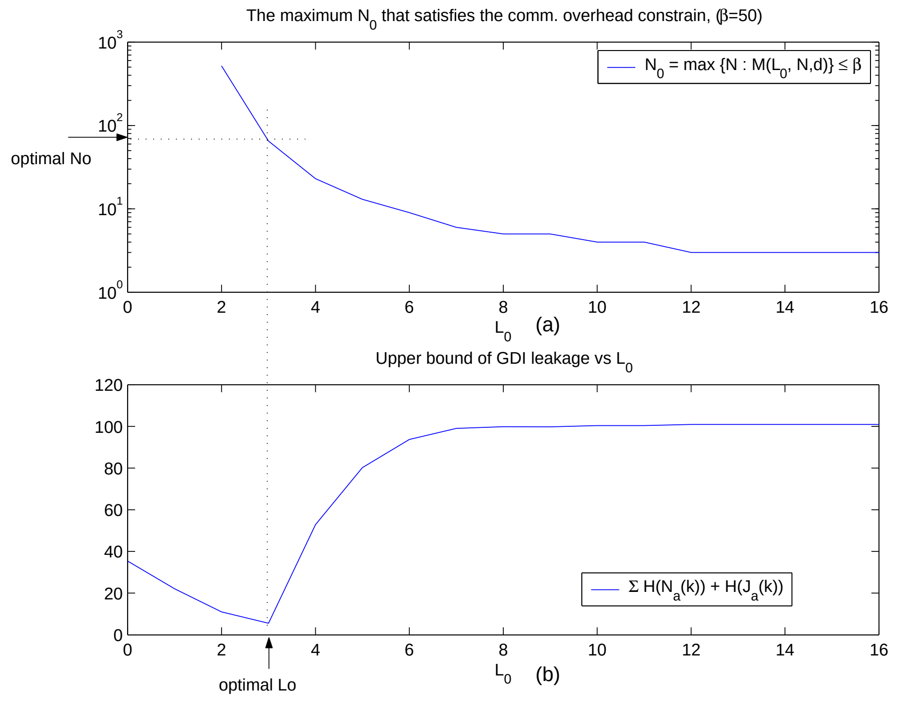


Figure 4.6: Illustration of selecting optimal parameters L_0 and N_0.

leakage. This figure demonstrates the upper bound of the mutual information as a function of the communication overhead constraint, where the parameters L_0 and N_0 have been optimized. This can help the system designer in determining the proper β for the communication constraint in (4.27). When not using phantom users, the artificial process is identical to the real process and we have $I(R; A) = I(R; R) = H(R)$. In this case, this particular multicast session require average 3.6 rekeying messages to be sent in every 15 minutes ($B_t = 15$) and has $I(R; A) \approx 137$. Figure 4.7 shows that the proposed anti-attack scheme can reduces $I(R; A)$ to 5.5 by increasing the communication overhead to 23.2 messages every 15 minutes. The communication overhead C_a is significantly larger than C_r because a large amount of activities of the phantom users must be created. However, the absolute value of the C_r is still small compared with the multicast data throughput. On the other

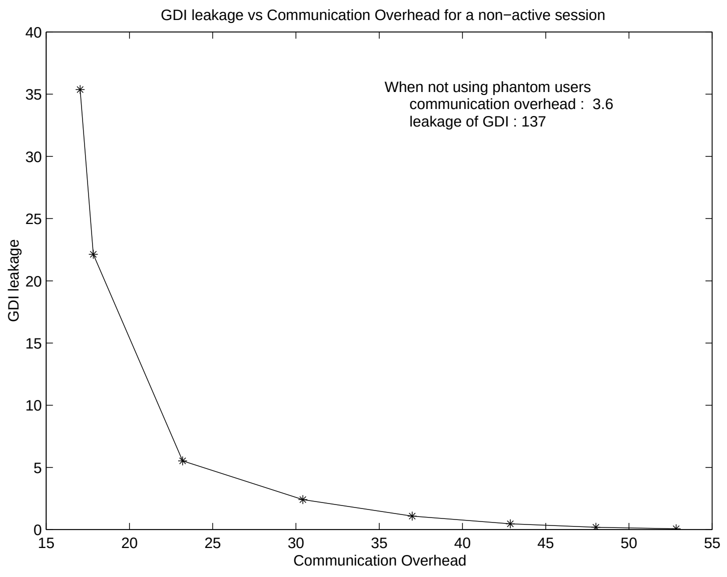


Figure 4.7: The GDI leakage versus communication overhead for a real MBone audio session

hand, the leakage of the group dynamic information is greatly reduced.

It is important to note that this MBone audio session contains only up to 60 users and represents the scenario where the group size is small and group members are not very active. Due to the lack of the experimental data for large multicast groups, we investigated a simulated multicast session with larger group size and more active group members. The simulation setup is the same as that is used for Figure 4.1(c) in Section 4.2, where the group size is about 500. When not using phantom users, the KDC sends average 28.16 rekeying messages in every 5 minutes ($B_t = 5$), while the amount of information leaked to the attackers, H(R), is 249.2. The performance of the proposed anti-attack methods is shown in Figure 4.8. We can see that the GDI leakage can be reduced to 5 at the expense of increasing the communication overhead to 93 messages per 5 minutes. The relative communication increase is smaller than that for the less active sessions.

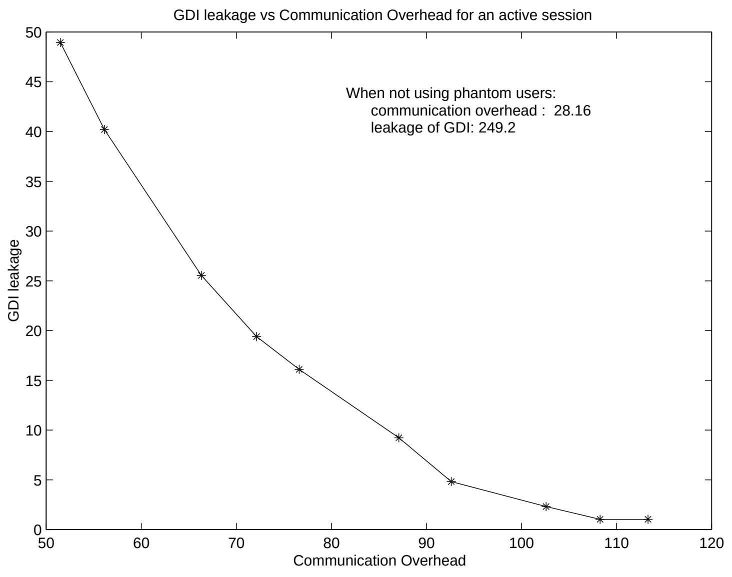


Figure 4.8: The GDI leakage versus communication overhead for a simulated multicast session

4.6 Contributory Key Management Schemes

As we have discussed in the previous sections, group keys are generated and distributed by the key distribution center when centralized key management schemes are employed. In many scenarios, however, it is not preferred to rely on a centralized server that arbitrates the establishment of the group key. This might occur in applications where group members do not explicitly trust a single entity, or there are no servers or group members who have sufficient resources to maintain, generate and distribute keying information. Thus, the distributed solution of the key management problem has seen considerable attention [18, 25–33].

The contributory key management schemes do not rely on centralized servers. Instead, every group member makes independent contribution and participates the process of group key establishment. The members' personal keys are not disclosed to any other entities [28]. An important class of contributory key management schemes, such as [25–33], are inspired by the Diffie-Hellman (DH) two-party key exchange protocol [35], and are usually refereed to as the *Diffie-Hellman-like* protocols. Compared with the centralized schemes, the contributory schemes have the advantage of not putting full trust on a single entity and therefore do not suffer the problem of single-point-failure. However, their distributed nature makes the task of protecting the GDI very difficult. In this section, we investigate the ways that the group members acquire GDI from the Diffie-Hellman-like key management schemes and provide a brief discussion on preventing the leakage of GDI.

4.6.1 Fully and partially contributory key management schemes

We discuss two slightly different flavors of contributory key agreement scheme: fully contributory and partially contributory.

In the fully contributory schemes, all key agreement operations are contributed to every group member [29]. Since there is no dedicated group manager, every participant may perform admission control and other administrative functions [29]. Thus, group members are naturally aware of the information about the group membership. In addition, group members are usually arranged in a logical ring [25], a logical chain [28–30], or a logical tree [31–33]. These logical ring/chain/tree structures describe the key establishment procedure, and must be maintained and updated by every member independently in the fully contributory environment [31]. This, again, requires members to have knowledge of the initial group membership as well as the membership changes. Therefore, the fully contributory schemes, whose implementation relies on the members' knowledge on dynamic group membership, are not suitable for the multicast applications with confidential GDI.

In the partially contributory schemes, one group member takes on a special role and performs some operations in a centralized manner [28] [29]. This special member is usually referred to as the *group controller*. The role of the group controller can be assigned to a fixed member or be handed over to other members when membership changes [29]. The group controller is different from the KDC in the centralized schemes because it does not hold the private keys of other members or generate the complete group key for other members. Instead, it may perform admission control and coordinate the process of the key formation. The original purpose of introducing group controller is to achieve efficient key updating in the case of user joining and departure [28]. In the context of protecting GDI, the partially contributory schemes make it possible to confine dynamic membership information to the group controllers while preventing other group members from accessing GDI. Although only a handful of contributory schemes [28–32] suggest using group controller, most of the schemes [25–27, 33] can work in the partially contributory manner. For contributory key management schemes, the fundamental rule for protecting GDI is that *a group controller that is trusted to handle GDI shall perform admission control, maintain the logical key ring/chain/tree structure and coordinate the process of the key formation.*

4.6.2 Vulnerability of popular contributory key management schemes

Utilizing group controller is not a complete solution to the GDI protection problem. Next, we examine the Diffie-Hellman-like key management schemes and demonstrate various other opportunities for the insiders to acquire group dynamic information.

The scheme presented in [25] is the earliest attempted to extend two-party Diffie-Hellman protocol to group scenario. This scheme, sometimes referred to as ING [30], arranges members in a logical ring and is executed in $(n-1)$ round, where n is the group size. Therefore, every member obtains the group size by simply counting the number of rounds that he performed.

Similarly, the schemes presented in [26] and [27], referred to as the STR and BD respectively, also reveal the group size. Here, each member receives the broadcast messages from all other members, and therefore must know the existence of other group members.

In [31] [32] [33], logical tree structures are introduced to manage the formation of the group keys. In these schemes, each member performs L rounds and holds L subgroup keys, where L is the depth of the key tree. Since L is proportional to the logarithm of the group size, group members know at least the order of the group size.

Another important set of contributory key management schemes are GDH.1, GDH.2 and GDH.3 [28]. These schemes arrange group members in a logical chain and accumulate the keying material by traversing group members one by one. In GDH.1/2, the k^{th} member receive k or $k+1$ messages from the $(k-1)^{th}$ member. Thus, the amount of the messages reveals information about the group size. The users who are closer to the end of the chain have more accurate information about the group size. GDH.3 is executed in four stages [28]. In the second and the fourth stage, the last user on the key chain broadcast n messages to the rest of the group, and n is the group size. In all three schemes, the group size information is revealed by the size of keying messages.

4.6.3 Prevention of GDI leakage

As discussed in Section 4.6.1 and 4.6.2, the contributory key management schemes are more vulnerable to GDI attacks than the centralized schemes. Besides examining rekeying message size as in the centralized schemes, the attackers can also steal GDI through performing admission control, maintaining the logical key ring/chain/tree structure, and counting the number of rounds in the contributory key management schemes.

In general, we suggest using centralized key management schemes for the applications with confidential GDI. However, there are scenarios that centralized schemes cannot be employed, such as when no trusted centralized entities exist. In these cases, we suggest using GDH.3, which has the strongest centrality flavor amongst contributory schemes. As discussed in Section 4.6.2, GDH.3 reveals group size through the broadcast message size. Thus, the following modifications must be made.

- Selecting the group member at the end of the logical chain as the group controller, who performs admission control and coordinates the key formation such that a regular member only communications with his two neighbors on the key chain and the group controller in the key establishment process.

- Replacing the broadcasting in the second and fourth stage [28] by multiple unicasting, which unfortunately increases the communication overhead.

The modified GDH.3 prevents regular group members from obtaining the information on the group size, at the expenses of non-scalable communication overhead. In addition, anti-traffic-analysis techniques, such as in [64] [65], shall be used to prevent the GDI attacks from outsiders, which will not be discussed in this work.

In this chapter, we raised the issues of the disclosure of dynamic group membership information through key management. We demonstrated that such a new security threat impacts various group communication applications. We developed attack strategies that could steal the GDI from key management. To protect GDI, an anti-attack framework was investigated, that involves utilizing batch rekeying, introducing phantom users, and analyzing the tradeoff between communication overhead and security.

Chapter 5

Conclusion and Future Work

This dissertation presented the design of network-specific and application specific group key management schemes and investigated the problem of protecting dynamic group membership information in secure group communications.

In particular, we presented a method for designing the multicast key management tree in the mobile wireless environment. By matching the key management tree to the cellular network topology and localizing the delivery of rekeying messages, a significant reduction in the communication burden associated with rekeying was observed compared to trees that are independent of the topology. We designed a topology-matching key management tree that consists of user-subtrees, BS-subtrees and SH-subtrees. It was shown that the problem of optimizing the communication cost for the TMKM tree is separable and can be solved by optimizing each of those subtrees separately. The ALX tree structure, which easily adapts to changes in the number of users, was introduced to build user-subtrees and BS-subtrees. The performance of the ALX tree is very close to the performance lower bound for any fixed degree tree. The GSHD algorithm, which considers the network heterogeneity where the SHs administer areas with varying network conditions, was introduced to build the SH subtree. The performance of the GSHD algorithm is very close to the optimal and has better performance than treating SHs equally. Additionally, we addressed the consequences that user mobility has upon the TMKM tree, and presented an efficient handoff scheme to reduce the communication burden associated with rekeying. A popular user joining/leaving procedure was used to study the performance of the TMKM and TIKM trees. Both simulations and analysis were provided. For systems consisting of only one SH, simulations performed for different user-join rates and mobile user speeds show that the cost of the TMKM tree is approximately 33-45% of the cost of the TIKM tree, which indicates a reduction of 55-67% in the total communication cost. For systems consisting of multiple SHs, simulations were performed for different amounts of participating SHs, and indicated that the TMKM tree can reduce the communication burden by as much as 80%. In addition, both analysis and simulations indicate that the communication cost of the TMKM tree scales better than that of topology-independent trees as the number of participating SHs increases.

While traditional group key management only provides the same access privilege to all group members, this dissertation presented a multi-group key management scheme that achieves hierarchical group access control in secure group communications. Hierarchical access control problem prevails in multimedia group applications, where multiple data streams are distributed to group members with various access privileges. We designed an integrated key graph, as well as the rekey algorithms, which allow users subscribing/dropping the group communications and changing access levels while maintaining the forward and backward security. Compared with using the existing tree-based key management schemes that are designed for a single multicast session, the proposed scheme can greatly reduce the overhead associated with key management. In the multi-layer services containing 4 layers, we observed more than 50% reduction in the usage of storage, computation, and communication resources in the centralized environments, and the number of rounds to establish and update keys in the contributory environments. More importantly, the proposed scheme scales better than the existing tree-based schemes, when the group applications contains more data streams and require the mechanism to manage more levels of access control.

Besides scalability issues, a more fundamental concern of group key management is security. This dissertation raised the issues of the disclosure of dynamic group membership information through key management in secure multicast communications. Such a security concern has not been addressed in traditional key management schemes. We demonstrated that the attackers can successfully obtain good estimates of the GDI from a large number of centralized and contributory key management schemes, and investigated the techniques of improving or modifying the existing key management schemes such that the GDI as well as the multicast content is protected. For the centralized key management schemes, we developed two effective attack strategies, which exploit the format and the size of the rekeying messages. To protect the GDI, we proposed the anti-attack technique utilizing batch rekeying and phantom users. This anti-attack technique reduces the leakage of the GDI and is fully compatible with the existing centralized key management schemes. We investigated the tradeoff between the communication overhead and the leakage of the GDI, and provided a framework for selecting the proper amount of phantom users. The proposed anti-attack technique was tested on real MBone user log data and simulated multicast sessions. We also demonstrated the vulnerability of the contributory key management schemes, where group members can acquire the group size from performing admission control, maintaining logical key ring/chain/tree structure, counting the number of rounds and measuring the number of the key exchange messages. In the contributory environment, the solution of protecting GDI involves utilizing the group controller and modifying the existing GDH.3 key establishment protocol.

Based on the research results presented in this dissertation, there are several research directions that can be further investigated:

Fault-tolerant contributory key agreement

Besides the technologies that have been presented in this dissertation, we plan to incorporate the fault-tolerate features in the future design, especially for contributory key management. The existing contributory key management schemes *assume that users honestly perform the key agreement protocol. As a result, they perform poorly or not at all in the presence of malicious group members who manipulate the keying messages and aim to cause the key agreement process to fails.* To demonstrate this fact, we examine one of the most popular schemes for distributed key agreement, namely GDH.2. This scheme organizes the users in a chain. Each user performs computation and passes some intermediate values to the next user. This stage of the scheme is referred to as upflow and continues until the computation and passing on of intermediate results has reached the last user at the end of the chain. Then, the direction of information flow is reversed and the scheme enters its downflow stage. The users once again perform computation on intermediate values and pass them on. During both upflow and downflow stages, malicious users have the opportunity to selectively sabotage the intermediate results calculated by the users that are ordered before them in the chain. At the end of the key agreement protocol, all users individually calculate the group key. The users, whom have been targeted by the malicious users, will have a different key from the rest of the group. In this case, no common group secret is established for securing group communication and the group key agreement protocol fails. For GDH.2, we can show that a malicious user can sabotage and remain undetected even after the key generation protocol is run many times with different orderings of the users. The failure of key agreement not only prevents secure group communication, but also causes extra use of computational and communication resources that are precious for wireless scenarios. All communication and computation spent on the key agreement is in vein since it is not possible to recover or reuse any part of previous group secret.

We believe that one of the largest problems with GDH.2 and any scheme that operates in a similar manner is the lack of verifying intermediate steps. In order to detect malicious users and recover from possible errors, our preliminary investigation suggests that distributed tree-based key agreement schemes, have the potential of detecting malicious users and allow for efficient recovery from key establishment failure. In tree-based key agreement schemes, the keys are generated recursively, allowing us to integrate detection and error recovery in the key generation process. We propose to design a protocol that detects malicious behavior/error by selectively validating the intermediate results and achieves fast recovery by reusing validated intermediate keys. The idea is to solve the dispute locally before it affects the entire group. We will investigate the tradeoff between the probability of successful establishing the group key and the increased communication and computation cost due to the checking mechanism.

Topology-aware hierarchical access control with GDI protection

In this dissertation, we have addressed the topology-aware key management, hierarchical access control and GDI protection separately. In the future, we plan to develop a suite of key management design techniques that incorporate all above advanced technologies. For example, topology-match key management has positive influence on protecting GDI because the users may only obtain their local GDI in TMKM scheme. In addition, the hierarchical key management design may also consider topology issues, when the access requirement is correlated with the physical locations of users.

More topics in wireless network security

Wireless communication has dramatically changed the way people work and interact. Unfortunately, the wireless era continues to be plagued by insufficient security. Key management solves the group access control problem, but it is not the complete solution of wireless network security problem. In wireless networks, the security weakness exists in every layer. In the future, we will investigate *secure routing in ad hoc networks* and *securing resource allocation against greedy users.*

Secure routing protocols are the foundation of the dependability in ad hoc networks. Various attacks, such as black/gray hole, rushing attack, blackmail, wormhole, prevent good routes being discovered and cause denial-of-service. We would like to develop a set of mechanisms to secure against routing disruptions. For each node, the first mechanism is to launch a route traffic observer to monitor the behavior of each valid route in its route cache, and to collect the packet forwarding statistics submitted by the hops on the routes. Since malicious nodes may submit false report, for each node, the next mechanism is to keep a cheating record database for the other nodes. If a node is detected as dishonest, future route discovery should prevent this node from being on the route. The third mechanism is to use friendship (trust relationship) to speed up the malicious node detection.

The fourth mechanism is to explore route diversity by discovering multiple routes to the destination, which can increase the chance to defeat the malicious nodes aiming to prevent good routes from being discovered. Instead of waiting for all the routes in its route cache becoming invalid, an adaptive route rediscovery mechanism is applied by each node to determine when a new route discovery should be initiated. Based on the observed behavior and the history record of each node, the design goal is to improve the network performance by limiting the damage of malicious attacks and detecting malicious nodes.

The common philosophy of resource allocation is to improve overall network performance or achieve fairness, by properly assigning network resources to users. One example is the power control algorithm in 3G wireless networks, where the base stations assign different transmission power to mobile users according to their channel conditions. Many resource allocation algorithms require the measurement or feedback of users' status, such as their computation capability and channel conditions. However, greedy or malicious users can make dishonest claims or manipulate the measurements, hoping that they can get more system resources or cause denial-of-service to other honest users. To make things even worse, it can be extremely difficult to detect these dishonest claims and manipulated measurements. I believe that resource allocation algorithms should consider this security threat in the early design stage. We propose to quantify this security concern and introduce a security constraint for resource allocation. There will be a trade-off between the optimality of the resource allocation and the robustness to false measurement/claims. The ultimate goal is to understand the interplay between security and quality of service.

As a summary, while attack and anti-attack as two major forces that drive the advancement of network security research, we believe that tomorrow will be better.

Appendix A

Calculation of $B(b, i, a)$

We define $n(b, i, a)$ to be the number of non-empty boxes when randomly placing i identical items into b identical boxes with repetition, where each box can hold at most a items. In this appendix, we calculate $B(b, i, a)$ as the expected value of $n(b, i, a)$, i.e. $B(b, i, a) = E[n(b, i, a)]$. It is obvious that the value of $n(b, i, a)$ is bounded as $B_0 \leq n(b, i, a) \leq B_1$, where $B_0 = \left\lceil \frac{i}{a} \right\rceil$ and $B_1 = \min(i, b)$.

We define an intermediate quantity $w(y, i, a)$ as the number of ways of putting i items into y boxes such that each box contains at least 1 and at most a items. $w(y, i, a)$ can be calculated recursively as:

$$\begin{aligned} w(B_0, i, a) &= \binom{aB_0}{i} && \text{(A.1)} \\ w(B_0 + k, i, a) &= \binom{a(B_0 + k)}{i} \\ &\quad - \sum_{m=0}^{k-1} \binom{B_0 + k}{B_0 + m} w(B_0 + m, i, a), && \text{(A.2)} \end{aligned}$$

where $0 \leq k \leq B_1 - B_0$. Then, the pmf of $n(b, i, a)$ can be expressed as:

$$Prob\{n(b, i, a) = B_0 + k\} = \frac{1}{N} \binom{b}{B_0 + k} w(B_0 + k, i, a), \tag{A.3}$$

where $N = \binom{ab}{i}$ represents the total number of ways of putting i items into b boxes. By substituting (A.2) into (A.3), we get:

$$\begin{aligned} Prob\{n(b,i,a) = B_0 + k\} = &\frac{1}{N}\binom{b}{B_0+k}\binom{a(B_0+k)}{i} \\ &- \sum_{m=0}^{k-1} \frac{\binom{b}{B_0+k}\binom{B_0+k}{B_0+m}}{\binom{b}{B_0+m}} Prob\{n(b,i,a) = B_0 + m\}. \end{aligned}$$

It can be shown that:

$$\frac{\binom{b}{B_0+k}\binom{B_0+k}{B_0+m}}{\binom{b}{B_0+m}} = \binom{b-B_0-m}{k-m}.$$

Therefore,

$$\begin{aligned} Prob\{n(b,i,a) = B_0 + k\} = &\frac{1}{N}\binom{b}{B_0+k}\binom{a(B_0+k)}{i} \\ &- \sum_{m=0}^{k-1} \binom{b-B_0-m}{k-m} Prob\{n(b,i,a) = B_0 + m\}. \end{aligned} \tag{A.4}$$

By substituting (A.1) into (A.3), we have:

$$Prob\{n(b,i,a) = B_0\} = \frac{1}{N}\binom{b}{B_0}\binom{aB_0}{i}. \tag{A.5}$$

Based on (A.4) and (A.5), we can calculate $Prob\{n(b,i,a) = B_0 + k\}$ for $k = 0, 1, \cdots, B_1 - B_0$ recursively. Then, we can calculate $B(b,i,a)$, i.e. $E[n(b,i,a)]$, as:

$$B(b,i,a) = \sum_{k=0}^{B_1-B_0} (B_0 + k) \cdot Prob\{n(b,i,a) = B_0 + k\}. \tag{A.6}$$

Appendix B

Calculation of pmf of $\widetilde{I}$

Let t_M denote the service duration, t_n denote the new cell dwell time, and t_h denote the previously handed-off cell dwell time [57]. We assume that t_M follows exponential distribution. The distributions of t_n and t_h are often presented together with the mobility models. For the mobility model used in Section 2.7, the distribution of t_n and t_h can be found in [57].

Using these distributions, we calculate $p_n = Prob\{t_M < t_n\}$ and $p_h = Prob\{t_M < t_h\}$. The number of cells that a user ever visited before departure, denoted by $\widetilde{I}$, has the pmf as $Prob\{\widetilde{I} = 1\} = p_n$, $Prob\{\widetilde{I} = 2\} = (1 - p_n)p_h$, $Prob\{\widetilde{I} = 3\} = (1 - p_n)(1 - p_h)p_h$, and $Prob\{\widetilde{I} = i\} = (1 - p_n)(1 - p_h)^{i-2}p_h$.

BIBLIOGRAPHY

[1] A. Perrig and J. D. Tygar, *Secure Broadcast Communication: In Wired and Wireless Networks*, Kluwer Academic Publishers, 2002.

[2] S. Paul, *Multicast on the Internet and its applications*, Kluwer Academic Publishers, 1998.

[3] M.J. Moyer, J.R. Rao, and P. Rohatgi, "A survey of security issues in multicast communications," *IEEE Network*, vol. 13, no. 6, pp. 12–23, Nov.-Dec. 1999.

[4] D.M. Wallner, E.J. Harder, and R.C. Agee, "Key management for multicast: issues and architectures," Internet Draft Report, Sept. 1998, Filename: draft-wallner-key-arch-01.txt.

[5] R. Gennaro and P. Rohatgi, "How to sign digital streams," in *Advances in Cryptology-Crypto '97*, 1997.

[6] C. K. Wong and S. Lam, "Digital signatures for flows and multicasts," *IEEE/ACM Trans. On Networking*, vol. 7, pp. 502–513, 1999.

[7] P. Rohatgi, "A compact and fast hybrid signature scheme for multicast packet authentication," in *6th ACM Conference on Computer and Communications Security*, 1999, pp. 93–100.

[8] F. Bergadano, D. Cavalino, and B. Crispo, "Chained stream authentication," in *Selected Areas in Cryptography 2000,Waterloo, Canada*, August 2000.

[9] B. Briscoe, "Flames: Fast, loss-tolerant authentication of multicast streams," Technical report, BT research, 2000, http://www.labs.bt.com/people/briscorj/papers.html.

[10] A. Perrig, R. Canetti, J.D. Tygar, and D. Song, "The tesla broadcast authentication protocol," in *RSA Cryptobytes*, 2002.

[11] A. Perrig, R. Canetti, D. Song, and J.D. Tygar, "Efficient and secure source authentication for multicast," in *Network and Distributed System Security Symposium, NDSS '01*, 2001.

[12] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar, "Spins: security protocols for sensor networks," in *Proceedings of the 7th Annual Conference on Mobile Computing and Networking (Mobicom)*, 2001, pp. 189–199.

[13] A. Perrig, J. D. Tygar, D. Song, and R. Canetti, "Efficient authentication and signing of multicast streams over lossy channels," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy (S&P 2000)*, 2000, p. 56.

[14] B. Briscoe and I. Fairman, "Nark: receiver-based multicast non-repudiation and key management," in *Proceedings of the 1st ACM conference on Electronic commerce*, 1999, pp. 22–30.

[15] S. Xu and R. Sandhu, "Authenticated multicast immune to denial-of-service attack," in *Proceedings of the 2002 ACM symposium on Applied computing*, 2002, pp. 196–200.

[16] R. Canetti, J. Garay, G. Itkis, D. Miccianancio, M. Naor, and B. Pinkas, "Multicast security: a taxonomy and some efficient constructions," *Proc. IEEE INFOCOM'99*, vol. 2, pp. 708–716, March 1999.

[17] C. Wong, M. Gouda, and S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. on Networking*, vol. 8, pp. 16–30, Feb. 2000.

[18] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The VersaKey framework: Versatile group key management," *IEEE Journal on selected areas in communications*, vol. 17, no. 9, pp. 1614–1631, Sep. 1999.

[19] W. Trappe, J. Song, R. Poovendran, and K.J.R. Liu, "Key distribution for secure multimedia multicasts via data embedding," *Proc. IEEE ICASSP'01*, pp. 1449–1452, May 2001.

[20] D. McGrew and A. Sherman, "Key establishment in large dynamic groups using one-way function trees," Technical Report 0755, TIS Labs at Network Associates, Inc., Glenwood, MD, May 1998.

[21] A. Perrig, D. Song, and D. Tygar, "ELK, a new protocol for efficient large-group key distribution," in *Proc. IEEE Symposium on Security and Privacy*, 2001, pp. 247 –262.

[22] G. H. Chiou and W. T. Chen, "Secure broadcasting using the secure lock," *IEEE Trans. Software Eng.*, vol. 15, pp. 929–934, Aug 1989.

[23] S. Mittra, "Iolus: A framework for scalable secure multicasting," in *Proc. ACM SIGCOMM '97*, 1997, pp. 277–288.

[24] S. Banerjee and B. Bhattacharjee, “Scalable secure group communication over IP multicast,” *JSAC Special Issue on Network Support for Group Communication*, vol. 20, no. 8, pp. 1511 –1527, Oct. 2002.

[25] I. Ingemarsson, D. Tang, and C. Wong, “A conference key distribution system,” *IEEE Transactions on Information Theory*, vol. 28, pp. 714–720, Sep. 1982.

[26] D. G. Steer, L. Strawczynski, W. Diffie, and M. Wiener, “A secure audio teleconference system,” in *Proceedings on Advances in cryptology.* 1990, pp. 520–528, Springer-Verlag New York, Inc.

[27] M. Burmester and Y. Desmedt, “A secure and efficient conference key distribution scheme,” *Advances in Cryptology- Eurocrypt*, pp. 275–286, 1994.

[28] M. Steiner, G. Tsudik, and M. Waidner, “Diffie-hellman key distribution extended to group communication,” in *Proceedings of the 3rd ACM conference on Computer and communications security.* 1996, pp. 31–37, ACM Press.

[29] M. Steiner, G. Tsudik, and M. Waidner, “CLIQUES: a new approach to group key agreement,” in *Proceedings of the 18th International Conference on Distributed Computing Systems*, May 1998, pp. 380 –387.

[30] M. Steiner, G. Tsudik, , and M. Waidner, “Key agreement in dynamic peer groups,” *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, vol. 11, no. 8, pp. 769–780, Aug 2000.

[31] G. Tsudik Y. Kim, A. Perrig, “Simple and fault-tolerant key agreement for dynamic collaborative groups,” in *Proceedings of the 7th ACM conference on Computer and communications security*, November 2000.

[32] L.R. Dondeti, S. Mukherjee, and A. Samal, “DISEC: a distributed framework for scalable secure many-to-many communication,” in *Proceedings of Fifth IEEE Symposium on Computers and Communications*, 2000, pp. 693 –698.

[33] W. Trappe, Y. Wang, and K.J.R. Liu, “Establishment of conference keys in heterogeneous networks,” in *proceedings of IEEE International Conference on Communications*, 2002, vol. 4, pp. 2201 –2205.

[34] K. Becker and U. Wille, “Communication complexity of group key distribution,” in *Proceedings of 5th ACM Conf. on Computer Commun. Security*, 1998, pp. 1–6.

[35] W. Diffie and M. Hellman, “New directions in cryptography,” *IEEE Trans. on Information Theory*, vol. 22, pp. 644–654, 1976.

[36] P. Judge and M. Ammar, "Gothic: A group access control architecture for secure multicast and anycast," in *Proceedings of the IEEE INFOCOM02*, 2002, p. 15471556.

[37] S.E. Eldridge and C.D. Walter, "Hardware implementation of montgomerys modular multiplication algorithm," *IEEE Transactions on Computers*, vol. 42, no. 6, pp. 693699, June 1993.

[38] A. Puri and T. Chen, *Multimedia Systems, Standards, and Networks*, Marcel Dekker Inc, 2000.

[39] Y. R. Yang, X. S. Li, X. B. Zhang, and S. S. Lam, "Reliable group rekeying: a performance analysis," *Proc. of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pp. 27 – 38, August 2001.

[40] C. Diot, B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the IP multicast service and architecture," *IEEE Network*, vol. 14, pp. 78 –88, Jan.-Feb 2000.

[41] A. Acharya and B.R. Badrinath, "A framework for delivering multicast messages in networks with mobile hosts," *Journal of Special Topics in Mobile Networks and Applications*, vol. 1, no. 2, pp. 199–219, Oct. 1996.

[42] H-S Shin and Y-J Suh, "Multicast routing protocol in mobile networks," *Proc. IEEE International Conference on Communications*, vol. 3, pp. 1416 –1420, June 2000.

[43] K. Brown and S. Singh, "RelM: Reliable multicast for mobile networks," *Computer Communication*, vol. 2.1, no. 16, pp. 1379–1400, June 1996.

[44] E. Ha, Y. Choi, and C. Kim, "A multicast-based handoff for seamless connection in picocellular networks," *Proc. IEEE Asia Pacific Conference on Circuits and Systems*, pp. 167 –170, Nov. 1996.

[45] Universal Mobile Telecommunications System (UMTS) Technical Specification, Digital cellular telecommunications system (Phase 2+ (GSM)), "Network architecture," 3GPP TS 23.002 version 5.9.0 Release 5, 2002-12.

[46] J.E. Wieselthier, G.D. Nguyen, and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," *Proc. IEEE INFOCOM'00*, vol. 2, pp. 585 –594, March 2000.

[47] L. Gong and N. Shacham, "Multicast security and its extension to a mobile environment," *Wireless Networks*, vol. 1, no. 3, pp. 281–295, 1995.

[48] M. Hauge and O. Kure, "Multicast in 3G networks: employment of existing IP multicast protocols in umts," in *Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*. 2002, pp. 96–103, ACM Press.

[49] "Mlisten," available at www.cc.gatech.edu/computing/Telecomm.mbone.

[50] K. Almeroth and M. Ammar, "Collecting and modeling the join/leave behavior of multicast group members in the mbone," in *Proc. High Performance Distributed Computing (HPDC'96), Syracuse, New York*, 1996, pp. 209–216.

[51] K. Almeroth and M. Ammar, "Multicast group behavior in the internet's multicast backbone (MBone)," *IEEE Communications*, vol. 35, pp. 224–229, June 1999.

[52] G.K. Zipf, *Human Behavior and the Principle of Least Effort*, Addison-Wesley Press, 1949.

[53] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, Addison Wesley, 2nd edition, 1994.

[54] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley-Interscience, 1991.

[55] M. Rajaratnam and F. Takawira, "Nonclassical traffic modeling and performance analysis of cellular mobile networks with and without channel reservation," *IEEE Trans. on Vehicular Technology*, vol. 49, no. 3, pp. 817–834, May 2000.

[56] M. Sidi and D. Starobinski, "New call blocking versus handoff blocking in cellular networks," *Proc. IEEE INFOCOM '96*, vol. 1, pp. 35–42, March 1996.

[57] M. M. Zonoozi and P. Dassanayake, "User mobility modeling and characterization of mobility patterns," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7, pp. 1239–1252, Sep. 1997.

[58] M. Eltoweissy and J. Bansemer, "A framework for scalable multicast security with bell-lapedulla confidentiality model," *Journal of Internet Technology: Special Issue on Network Security*, July 2002.

[59] D. Bell and L. La Padula, "Secure computer systems: Mathematical foundations and model," in *MITRE Report, M74-244, MTR 2547 v2*, Nov. 1973.

[60] B. Sun, W. Trappe, Y. Sun, and K.J.R. Liu, "A time-efficient contributory key agreement scheme for secure group communications," *Proc. of IEEE International Conference on Communication*, vol. 2, pp. 1159 –1163, 2002.

[61] "http://ftp.cc.gatech.edu/people/kevin/release-data," .

[62] Y. Amir, G. Ateniese, D. Hasse, Y. Kim, C. Nita-Rotaru, T. Schlossnagle, J. Schultz, J. Stanton, and G. Tsudik, "Secure group communication in asynchronous networks with failures: Integration and experiments," in *Proceedings of IEEE ICDCS 2000*, April 2000.

[63] Y. Sun, W. Trappe, and K.J.R. Liu, "An efficient key management scheme for secure wireless multicast," *Proc. of IEEE International Conference on Communication*, vol. 2, pp. 1236–1240, 2002.

[64] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous connections and onion routing," *IEEE journal on selected areas in communications*, vol. 16, pp. 482–494, May 1998.

[65] R.E. Newman-Wolfe and B.R. Venkatraman, "High level prevention of traffic analysis," in *Proceedings of Seventh Annual Computer Security Applications Conference*, Dec. 1991, pp. 102–109.