ABSTRACT

Title of Dissertation: Multi-User Security: A Signal Processing and Networking Perspective

Wade Trappe, Doctor of Philosophy, 2002

Dissertation directed by: Professor K. J. Ray Liu
Applied Mathematics and Scientific Computing
Department of Electrical and Computer Engineering
Institute for Systems Research

The advancements in communication and multimedia technologies have paved the way for a new suite of multi-user applications that will allow users to interact. Although the new communication infrastructure makes it easier to reach the end user, it also makes it easier for adversaries to mount attacks against security measures intended to protect data. Thus, there must be mechanisms in place that guarantee the confidentiality and rights of both the customer and the service provider during the delivery of content across future communication networks.

This thesis examines security issues related to communications involving more than two participants or adversaries. We approach the problem of multi-user security by developing security measures at different stages of the content distribution

process, ranging from the establishment of initial keying information before transmission, to key management while delivering through networks, and finally to content protection and collusion prevention/tracing after delivery.

We address the issue of establishing a group key prior to content delivery by introducing the butterfly scheme and a conference keying scheme that addresses user heterogeneity. These schemes employ the two-party Diffie-Hellman scheme in conjunction with an underlying algorithmic tree called the conference tree. In order to address client heterogeneity, we design the conference tree using source coding techniques to account for the different user cost and budget profiles. We also introduce the PESKY performance measure, which quantifies the likelihood that a conference key can be established in a heterogeneous environment.

We then consider the problem of managing keys during content delivery by proposing a multicast key management system that uses a composite message format with member join and departure operations. Compared with the traditional format of the rekeying messages used in tree-based multicast key management, our composite message format reduces the amount of header information, while maintaining the same payload size.

Finally, we address the issue of protecting the digital rights of multimedia content after it has left the protected or encrypted domain. Since traditional multimedia fingerprints are susceptible to collusion attacks made by a coalition of adversaries, we develop fingerprints for multimedia that are based upon code modulation and able to identify groups of colluders.

Multi-User Security: A Signal Processing and Networking Perspective

by

Wade Trappe

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
2002

Advisory Committee:

Professor K. J. Ray Liu, Chairman/Advisor
Assistant Professor Min Wu
Professor Lawrence Washington
Professor Dennis Healy
Professor Prakash Narayan
Professor Virgil Gligor

© Copyright by

Wade Trappe

2002

DEDICATION

To my parents, who have each guided me and given me inspiration to face life's challenges. This thesis and my education would not have been possible without their sacrifices and the encouragement they gave throughout my youth.

ACKNOWLEDGEMENTS

"Few can foresee whither their road will lead them, till they come to its end."
I started graduate school with a plan– a very narrow and focused plan that even consisted of a thesis topic. But at many points on the road, life introduced some perturbations into my grand plan that yielded trajectories exponentially separated from the original path I had laid out. Many of these curve balls led to uncharted territories that taught me painful and valuable lessons about more than just science and math, but about life. I do not think that I could have made it through these wild and dark places if it were not for the help and love of many people who were there for me. As I sit here at my laptop wrapping up the final pages of my dissertation, the culmination of an arduous journey, it is with great emotion that I acknowledge their help.

I'd like to begin with my family. Both of my parents have, through the many years, given me encouragement to seek knowledge. But more than that, they have each given me their love. My mom has been there to teach me the value of patience, and in many invisible ways led me on my path of spiritualism. My dad has helped me with his wealth of experience about graduate school, which has shown me the timeless universality of my struggles. Tara, my sister, deserves much gratitude and apologies for putting up with a grumpy brother who has often been too busy to call home. I have always admired her diligence and her ability to check in on

me. Finally, my grandparents, Ralph and Donna, were there for me when I started school, and sacrificed much for my sister and I. Their memory has inspired me, and will always guide me.

I was fortunate enough to have found a mentor who was very well suited to guiding and advising me throughout my studies. Professor Liu provided wisdom that far surpassed anything that I could learn from books. It was through his unique way of challenging me that I chose to leave the path I had planned. From the moment that I gave up my almost-religious loyalty to wavelets, the journey became tougher. The rewards, however, have been far greater than I could have expected. I look back and realize that I have gained more than just a degree, but a *philosophy*, which is what this journey ultimately was about.

One unexpected and enjoyable fork in the road came from my late night discussions with Professor Washington during my first year of graduate school. Developing our cryptography class was a lot of fun (and a lot of hard work) that finally paid off. My research into security would not have happened if we had not developed that class and written our book. He was a helpful mentor during my first year, and has been a valuable colleague throughout my graduate research.

There are colleagues who I have had the pleasure of working with and deserve some recognition for helping me in my development. I was very fortunate that Min Wu decided to come to the University of Maryland during my last year of graduate studies. The work presented in Chapter 5 was done in collaboration with Min and could not have been done in such a timely manner if it had not been for the friendship that we experienced. At many points during our stint of sixteen hour work days, camaraderie kept both of us going. She deserves the credit for the simulations involving ACC and images, which were done using her

MATLAB code. Further, the idea of using divide and conquer to improve the computational efficiency of performing orthogonal signal detection was proposed by Min. The works presented in Chapter 3 and Chapter 4 were done in collaboration with Jie Song and Radha Poovendran while they were both at the University of Maryland. In particular, the simulation results for data embedding were done by Jie Song. I will fondly remember the hours spent at the whiteboard with Radha during the initial stages of our work on multicast key management. The work presented in Chapter 2 was done in collaboration with Professor Yuke Wang of The University of Texas at Dallas, who provided a refreshing viewpoint to conference key establishment that ultimately led to the butterfly scheme. I also would like to thank my committee members for their valuable feedback which has improved the quality of this thesis.

There are a few events in life that convince you of the existence of fate. It was just a few years ago when the most fortunate event of my life happened- when my path crossed that of an old friend, and our lives came together. Nisha was the ray of sunshine that pushed back the gloom and guided me through the last half of graduate school. To my wife, I would like to say a special thank you for everything. It is with heartfelt emotion that I will submit this thesis and begin a new phase of our life together.

I was also fortunate enough to have a very close friend who helped me out and called me more than once during my hard times. She has been Castor to my Pollux, and has listened to my troubles when things were hard for me, while I consoled her through her own hard times. It is a coincidence that we both are finishing our degrees this Spring 2002 semester, but perhaps also a statement of our close friendship. To Sheilagh, I would like to send my gratitude and congratulations.

When this journey started, I left Austin listening to a song by a little-known, and now-dissolved group called Fabu. I did not realize the significance of the lyrics "I will walk to the sea and everything shall remind me of you", but in reflection those words have become clear now. I left my coterie in Austin, and the separation was difficult. Having left the comfort of home for the uncertain promises of the East, I can look back now and truly appreciate the value of the friendships I had. Some part of this thesis is for my coterie: Sheilagh, Tina, Shayna, Steve (who's national champs now?), Tracy, Kendall, and Medha. As we continue to grow, may we maintain our friendships wherever our roads take us.

While in Maryland I have made many close friendships. I am fortunate that I gained a close friend in Ioannis. It was beyond my dreams that he would come to India for my wedding, or that he would put up with me for so many years as a roommate. I also was lucky to meet Steve and Danielle Levitt during the beginning of my studies. Although its been hard to keep in touch with them, their friendship has been deeply felt. I have also had exceptional friends in my research group who have had to put up with my eccentricities. Particularly, I want to acknowledge Haitao, Javad, Masoud, Alejandra, Xiaowen, and Nitin. To Zoltan (who gets special thanks for epsilon and that marginal operations research approach) and Yan, I pass on the torch and extend my best wishes– you guys made the homestretch fun. I have lived with some of the best roommates imaginable. I want to extend thanks to Josh, Ioannis, Jim, Karen, Jake, Ken and Isaac for the years we've shared. A special part of my life in Maryland has been shared with my wife's family. Both Daya and Madhu Gilra have treated me as part of the family, given me advice during the hard times, taught me how to cook Indian food, and kept me going with delicious dinners. Finally, I would like to thank everyone else

in Maryland who has shared the journey with me.

On a personal front, I want to acknowledge two final people who helped me in rather unique ways. Several years ago I met Sharon Zwillinger, who did more to help me through my toughest trials than I am sure she is aware of. Her kindness and guidance helped me to understand myself, and find the personal balance that I needed. I also found comfort from another source– the music of Dar Williams. I cannot estimate how many hours I have played her songs and found understanding in their lyrics. Her lyrics kept me dancing when the music had ended.

Now, without further adieu I conclude my monologue and sheath my quill.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Several key technologies have matured in the past decade, allowing for the possibility of building new infrastructures for the delivery and consumption of data. On the application side, multimedia content has become ubiquitous. Content editing software and hardware, such as digital cameras, are allowing for users to easily create content. The availability of the Internet and the Web has encouraged artists, both professional and amateur, to share their creative expressions. Additionally, communication and networking technologies have advanced. The deployment of various broadband communication technologies, such as digital subscriber line (DSL) and fiber optical communications, has led to a rapid price drop for bandwidth. The result has been that the geographical barriers that used to hinder the communication between people has dissolved, and the Internet has become the ubiquitous, global network for one to be in touch anywhere, anytime. This combination of application and communication technologies is creating opportunities for new businesses to meet the growing global demand for information and entertainment.

As users are brought *virtually* closer to each other and made increasingly aware of each other, they will want to interact. Whether for good or for bad purposes, they will be able to share experiences that allow them to work or play together.

Already, new commercial markets, such as interactive television and mobile video conferences, are on the horizon and promise to take advantage of the available bandwidth. It is no longer difficult to envision a future where users will personalize their experiences by interacting with the multimedia content and combining different media streams from different content providers.

However, after a decade of intensive development of both networking and multimedia technologies, the barrier to the successful deployment of the next generation of infotainment services no longer lies with bandwidth-related issues, but with assuring that content is used for its intended purpose by its intended recipients. The core issue is thus the secure management of content usage and delivery across the future ubiquitous and heterogeneous networks.

The recent controversy between the music industry and software/Internet companies has only emphasized the importance that this issue will play in the future. Programs like Napster and Gnutella have made the distribution of MP3 audio files relatively simple, which has angered the record labels since such software circumvents the conventional sources for revenue, and does not provide an alternative mode of remuneration. The need to create alternative methods to sell and distribute content on the future communication networks has spurred several international groups to investigate flexible solutions to rights management and content access. Two key examples already exist in the industry. First, is the secure digital music initiative (SDMI), which was created to develop technologies that would enable a new market for digital music to emerge by providing convenient access to content while supporting copyright protection for the artists' work [1]. Second, ISO/IEC Working Group 11, more commonly known as MPEG, foresees similar problems arising for the distribution of video and other multimedia formats. Al-

ready, MPEG is investigating intellectual property management and protection (IPMP) solutions under the framework of MPEG-4 [2] and MPEG-21 [3].

Due to the advancements in communications, the problem of controlling access to data and protecting the digital rights of content is no longer an issue that is able to be addressed solely through conventional cryptographic paradigms. No longer will the market only demand the traditional two-party application where a single consumer purchases unique data or content from a service provider. Instead, there will be an increased demand for applications that involve the concurrent distribution of content to multiple users. Although the new communication infrastructure makes it easier to reach the end user, the nature of these ubiquitous networks also make it easier for adversaries to observe transactions, collect data, and mount attacks against security measures intended to secure data and content distribution. Further, it is easier for adversaries to work together and combine their separate resources to subvert security measures. Therefore, there is a need for security measures that control the access to content when it is distributed to multiple users, as well as guaranteeing the digital rights of the content in the presence of coalitions of adversaries seeking to use the content for illicit purposes.

1.1 Thesis Overview and Contributions

In this thesis, we examine *multi-user security* for applications that will occur on the networks of the future. Multi-user security, in short, is concerned with the security issues related to communications involving more than two parties. This definition is, intentionally, vague and meant to incorporate a variety of different scenarios. In particular, multi-user security covers the problem of providing confidentiality during the distribution of data to more than two users. It is, however,

not restricted to apply to multiple service participants, but may also apply to maintaining security against multiple adversaries. Therefore, we may summarize multi-user security as:

Definition 1. *Multi-user security is the collection of issues related to guaranteeing the confidentiality, integrity, authenticity, and appropriate usage of data that is distributed by a group of senders to a group of receivers in the presence of a group of adversaries.*

In this definition, there are three different classes of entities: senders, receivers, and adversaries. It is possible that some of these entities might overlap. For example, in conference applications, a user might send data to all other members of the conference, and himself receive data from every other member of the conference. It could also be possible that an adversary might pose as a member of a service for a short time in order to acquire data with which to impersonate other members at a future time. Further, in each of the three classes, it may be that a group consists of only one member or multiple members.

Ensuring the secure management of data and content usage by multiple users on heterogeneous networks requires new security solutions that address the plethora of challenges posed by the diverse clientele and network profiles. In contrast to the traditional security paradigm, where security is treated as a layer of a system that is separated from network-related and application-specific issues that might have an effect on the performance of the security measures, we consider the influence of the communication network and the specific application on the issues of security. This thesis considers that the security system must be designed and tailored for the specific application or communication network. To address the role that the network plays on a security system, we examine the formation of keying information

needed to secure a group of users with different communication or computational capabilities. We also present security solutions that are designed specifically for multimedia applications.

The problem of multi-user security is approached in this thesis by developing security measures at different stages of the content distribution and consumption process. In particular, we consider the complete chain of elements, ranging from the establishment of initial keying information before transmission, to key management while delivering through networks, and finally to content protection and collusion prevention/tracing after delivery. In the subsections that follow, we describe the contributions of the thesis in each stage of the content delivery and consumption process.

1.1.1 Prior to Content Delivery

Prior to the delivery of group data, it is necessary to initially establish keying material used to secure the data in a group application. Key distribution 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 that they use to agree upon a key. When initial group session keys are being formed by a centralized entity, traditional cryptographic protocols can be used to distribute the key material to the users.

However, in many application environments, it is not possible to have a third party distribute the group key. This might occur in applications where the group members do not have sufficient confidence in any single entity's trustworthiness to perform key distribution. In this case, it is necessary to have the group members each make contributions to the formation of the group key. *Contributory* protocols,

where each member participates in the formation of the group key, are known as *conference keying* schemes.

Typical conference key establishment schemes seek to minimize either the amount of rounds needed in establishing the group key, or the size of the messages, and treat all users as identical. Group-oriented services will ultimately be heterogeneous in nature, bringing together users with varying amounts of computing power and communication capabilities. Therefore, in order to secure tomorrow's communication systems, it is essential to develop security solutions that address the heterogeneous nature of an application's clientele. Since many applications will involve a heterogeneous clientele consisting of group members with different computational capabilities, pricing plans, and bandwidth resources, minimizing the total bandwidth or the amount of rounds might not be an appropriate performance metric to optimize. Instead, one should aim to minimize a cost function that incorporates the different costs and capabilities of each user. Further, when users have resource constraints imposed upon them, the key generation procedure must decide whether it is feasible to generate a key and determine a procedure for generating the group key while minimizing the total cost subject to resource budget constraints.

In Chapter 2, we introduce the butterfly scheme, a scalable method that constructs the conference key using the two-party Diffie-Hellman scheme as the basic building block. The development of the butterfly scheme was inspired by the butterfly communication diagrams of FFT computations, which efficiently distributes input information amongst the processing nodes. Underlying the butterfly scheme is a tree, called the conference tree, that describes the successive subgroups and subgroup keys that are formed en route to establishing the key for the entire group.

The use of tree-based conference key establishment reduces the amount of rounds needed to establish the conference key from being linear, as is the case in [4, 5], to being logarithmic in the group size.

The butterfly scheme treats the group as homogeneous, and does not consider that users have different profiles. In order to address client heterogeneity, we propose to tailor the design of the conference tree to account for the different user cost and budget profiles. We assume that each user has a cost associated with performing one two-party DH scheme, or a budget that describes the amount of two-party DH schemes he is willing to participate in. We may seek conference trees with small average user cost by choosing the conference tree as the tree associated with coding a source with symbols whose weights are the different user costs. Source coding techniques, such as Huffman coding, allow for the design of conference trees that minimize the average user cost. In some scenarios users might have budget constraints, in which case the necessary conditions for the ability to generate a tree-based conference key is that the vector of user budgets satisfy the Kraft Inequality. In order to compare our heterogeneous tree-based scheme with other conference keying schemes, we introduce the PESKY measure (the probability of establishing the session key). PESKY quantifies the likelihood that a conference key can be established for a heterogeneous clientele of users whose budgets are drawn according to an underlying probability distribution. The PESKY of our tree-based schemes was found to be larger than the PESKY for other schemes when the user budget distribution did not have a single entity with significantly more resources than the other users. Further, we study the effect of falsely announced costs on the design of the conference tree, and propose the usage of a clipping operator with a threshold determined by minimizing the divergence

in order to reduce the effect of miscoding when designing the tree.

1.1.2 During Content Delivery

The most appropriate communication paradigm for the delivery of content to multiple users is multicast communications. This technology will be critical for bandwidth intensive services, such as video-on-demand and pay-per-view, where users will simultaneously enjoy the same content. Many of these applications will involve dynamic group scenarios, where users may join and leave at anytime, which makes it necessary to update the data needed to maintain the integrity of a multicast application's security. A conditional access system for multicasting must be able to make necessary changes to the encryption keys that protect a service. The most appropriate framework for handling server-oriented content distribution is by using a centralized entity that is responsible for maintaining the integrity of the users' keys.

Many schemes have been proposed to provide key management for server-based group applications, though the most common class of multicast key management schemes employ a tree hierarchy of keys [6–8]. Tree-based schemes are scalable since the amount of communication and storage resources needed does not become a hindrance to providing the service as the multicast group membership increases.

A deficiency with the current multicast key management schemes that have been proposed is that their design has primarily focused on the issue of reducing the size of the payload (the rekeying information), and not on the size of the entire message (including the rekeying message and the header). In fact, the transmission of the messages that flag the users which portion of the message is intended for them is an essential element of an application that can add significant communication

overhead when used in conventional tree-based schemes.

In Chapter 3, we examine the problem of managing keys for applications that multicast multimedia data. We provide an overview of fundamental information theoretic results related to securely multicasting information to a group of privileged users. This overview provides motivation and background for reducing the communication cost associated with multicast key distribution. We then introduce multicast key management, and discuss different methods for conveying the rekeying messages. In particular, multimedia data provides a media-dependent channel, accomplished through data embedding techniques, for conveying the rekeying messages. The primary advantage of using the media-dependent channel to convey rekeying messages compared to the traditional use of a media-independent channel is that the data embedding channel hides the presence of rekeying messages from potential adversaries, thereby making it more difficult for eavesdroppers to measure information regarding membership dynamics. Further, the use of data embedding allows the application to maintain the data rate of the media without performing computationally expensive transcoding operations. We next introduce a new format for the rekeying messages associated with departures from the group membership. This new message format, which we call the *residue-based* format, provides a single *homogenized* message from which each user can extract the new key, and does not require the usage of header information to flag the users to their portion of the message. We provide an analysis of the security of the message format, and show that the difficulty for internal adversaries, who are members of the group service, to acquire the keys of other users is atleast as difficult as that of breaking the cryptographic primitive of one-way functions. We present simulations illustrating the feasibility of performing rekeying when using data embedding

in H.263 video. Finally, we show that by adding extra functionality to multiple key trees, multicast key distribution schemes can be extended to protect multiple layers of multimedia content in an efficient manner.

In Chapter 4, we further investigate the usage of a homogenized message format by proposing a multicast key management system that uses a composite message format with member join and departure operations. This system can incorporate any composite message format, such as the one in Chapter 3. The analysis presented uses a message format based upon polynomial interpolation [9], where a single message consists of information needed to build a polynomial from which each user can calculate the new key. Compared with the traditional format of the rekeying messages used in tree-based multicast key management, our composite message format reduces the amount of header information, while maintaining the same payload size. Further, we optimize the parameters associated with the design of the tree by introducing a stochastic population model where each leaf node is occupied according to i.i.d. Bernoulli random variables. Under this occupancy model, we show that the average case communication requirements are close to the worst-case communication requirements, and we optimize the tree for the worst-case.

1.1.3 Post-Content Delivery

Although access control is an essential element to ensuring that content is used by its intended recipients, it is not sufficient for protecting the value of the content. The protection provided by encryption disappears when the content is no longer in the protected domain. It is therefore necessary to provide mechanisms that protect content usage after it is delivered. In order to control the redistribution of con-

tent, digital fingerprinting is used to trace the consumers who use their content for unintended purposes [10, 11]. These fingerprints can be embedded in multimedia content through a variety of watermarking techniques [12, 13]. Conventional watermarking techniques are concerned with robustness against a variety of attacks such as filtering, but do not always address robustness against a coalition of users with the same content that contains different marks. These multi-user attacks, known as *collusion* attacks, can provide a cost-effective approach to removing an identifying watermark.

In Chapter 5, we investigate the problem of making fingerprints for multimedia content, such as images and video, that can withstand collusion and allow for the identification of colluders. We begin by introducing the collusion problem for additive embedding. We show that under reasonable assumptions, the optimal fair strategy for colluders performing an averaging attack is to perform an average where each user weighs their marked content equally. We then study the effect that collusion has upon orthogonal, biorthogonal, and simplex modulation. We introduce an efficient detection algorithm for identifying the fingerprints associated with K colluders that requires $\mathcal{O}(K \log(n/K))$ correlations for a group of n users.

We next develop a fingerprinting scheme based upon code modulation that does not require as many signals as orthogonal or simplex modulation in order to handle n users. Our fingerprints are based upon anti-collusion codes (ACC), which have the property that the composition of any subset of K or fewer codevectors is unique. Using this property, we may therefore identify groups of K or fewer colluders. We present a construction of binary-valued ACC under the logical AND operation that uses the theory of combinatorial designs and is suitable for both the on-off keying and antipodal form of binary code modulation. In order to accommodate n users,

our code construction requires only $\mathcal{O}(\sqrt{n})$ orthogonal signals. For practical values of n, this is an improvement over prior work on fingerprinting generic digital data. We demonstrate the performance of our ACC for fingerprinting multimedia and identifying colluders through experiments using Gaussian signals and real images. In our study, we observe a close interplay between the detector and the desired ability to capture colluders as well as the unwanted side-effect of placing innocent users under suspicion.

Chapter 2

Conference Key Establishment for Heterogeneous Networks

Prior to the delivery of data intended for a group of recipients, it is necessary to initially establish keying material used to secure the group application. In this chapter we investigate the initial key agreement problem for both homogeneous and heterogeneous networks, whereby the members of a group each make contributions to establishing secret information that may be used to form a group encryption key.

2.1 Introduction

The advancement of communication technology is leading to a future where group-based applications will become a reality. Many applications will require that the communication amongst group members be protected from unwanted eavesdroppers. Corporate conferences, with members from different parts of the world, might contain industrial secrets that are in the best interests of the corporation to keep unknown to rivals. In order to protect the communication traffic, the information

must be encrypted, requiring that the privileged parties share an encryption and decryption key. Key distribution 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 that they use to agree upon a key.

In the centralized approach to group key establishment, either a group leader or a trusted third party is responsible for the generation and distribution of keying material. The problem of centralized group key distribution has seen considerable attention recently in the literature [6, 8, 14]. In many cases, however, it is not possible to have a third party arbitrate the establishment of a group key. This might occur in applications where group members do not explicitly trust a single entity, or no member has the resources to maintain, generate and distribute information by himself. In these cases, *contributory* approaches are needed, where the group members each make independent contributions to the formation of the group key.

The classic example of a contributory scheme is the Diffie-Hellman (DH) key establishment scheme [15], in which two parties exchange messages that allow them to securely agree upon a key that may be used to protect their two-party communication. Several researchers have studied the problem of establishing a *Diffie-Hellman like* conference key [4, 5, 16–19]. Typically, these conference key establishment schemes seek to minimize either the amount of rounds needed in establishing the group key, or the size of the message. Many applications, however, will involve a heterogeneous clientele consisting of group members with different computational capabilities, pricing plans, and bandwidth resources. For these applications, minimizing the total bandwidth or amount of rounds might not be an appropriate metric. Instead, one should aim to minimize a cost function that incorporates the different costs or resource constraints of each user. The key generation

scheme must therefore decide whether it is feasible to generate a key and determine a procedure for generating the group key while minimizing the total cost subject to resource budget constraints.

In this chapter, we develop methods for efficiently establishing a Diffie-Hellman like conference key that address the heterogeneous requirements of the conference members. We start in Section 2.2 by reviewing the Diffie-Hellman protocol, and presenting several conference keying schemes that employ the Diffie-Hellman problem. In Section 2.3, we present the butterfly scheme, a conference keying scheme for a homogeneous group of users, which builds the group key using the approach of [4]. The butterfly scheme can be generalized and we show that an underlying tree, which we call the *conference tree*, governs the process by which subgroup keys are formed en route to establishing the group key. By examining different shapes of conference trees, a family of tree-based group DH schemes can be formed. In Section 2.4, we consider the problem of designing a conference tree when the users have different capabilities. We first examine the case when the users have different costs. In this case, the optimal conference tree can be constructed using the Huffman algorithm. We then examine the problem of choosing a conference tree when the users have the same cost, but are subject to varying budget constraints. We present necessary conditions for the existence of a conference tree when the users have budget constraints, and present an algorithm that minimizes the total cost given the budget constraints. Next, we consider the more general case where the users have different costs as well as different budgets. A computationally efficient near-optimal algorithm is presented that determines a conference tree whose total cost is very close to the optimal performance achieved by conference trees determined using either full-search or integer programming techniques. In Section 2.5,

we present the results of simulations comparing the cost of forming a group key using tree-based schemes and several existent schemes. We also present simulations comparing the likelihood that a group key can be formed given that the users' budgets are drawn according to different distributions. From these simulations we conclude that the tree formulation for establishing a group key allows for great flexibility, and can efficiently establish group keys in resource-limited scenarios. Finally, in Section 2.6, we study the effects that the quantization and clipping of user costs have upon the total cost, and then investigate the effect that untrusty users can have upon the total cost of forming the group key using the Huffman-based conference tree. By suitably choosing the appropriate threshold level in the clipping operator, the effects of miscoding are ameliorated. In Section 2.7, we summarize our results and present conclusions.

2.2 Group DH Overview

In the basic DH scheme, the operations take place in an Abelian group G, typically chosen to be $\mathbf{Z}_p$ (the integers mod a prime p), or the points on an elliptic curve under appropriate laws of addition [20]. For consistency of notation, we shall develop our results for the group $\mathbf{Z}_p$. A group element g is chosen such that g generates a suitably large subgroup of G (preferably the whole group). Both party A and party B choose a private secret $\alpha_j \in \mathbf{Z}_p^*$ where $j \in \{A, B\}$ and $\mathbf{Z}_p^*$ denotes the non-zero elements of $\mathbf{Z}_p$. They each calculate $y_j = g^{\alpha_j}$ and exchange y_j with each other. Party A then calculates the key via $K = (g^{\alpha_B})^{\alpha_A} = g^{\alpha_B \alpha_A}$ and similarly for party B.

The problem of establishing a *Diffie-Hellman like* conference key has been investigated by several others [4, 5, 16]. One of the first *Diffie-Hellman like* con-

ference key establishment schemes was proposed by Ingemarsson et al [4]. In the Ingemarsson (ING) scheme, the group members are arranged in a logical ring (e.g. $A \rightarrow B \rightarrow C \rightarrow A$). In a given round, every participant receives a message from its left-hand neighbor, raises that to their exponent, and passes it to their right-hand neighbor. For example, in the first round of a three person group exchange, we have $A \rightarrow B : g^{\alpha_A}$, $B \rightarrow C : g^{\alpha_B}$ and $C \rightarrow A : g^{\alpha_C}$. Then, in the second round $A \rightarrow B : (g^{\alpha_C})^{\alpha_A}$, $B \rightarrow C : (g^{\alpha_A})^{\alpha_B}$, and $C \rightarrow A : (g^{\alpha_B})^{\alpha_C}$. Finally, the shared key is $g^{\alpha_A \alpha_B \alpha_C}$, which they each can calculate by raising the final received message to their private exponent. For n users this scheme requires $n - 1$ rounds.

Another notable scheme is the Burmester-Desmedt conference key scheme [16]. This scheme consists of three rounds. During the first round, each user u_j generates a random exponent α_j and broadcasts $z_j = g^{\alpha_j}$. The second round consists of each user u_j receiving z_j and broadcasts the quantity $x_j = (z_{j+1} z_{j-1}^{-1})^{\alpha_j}$. In the final round, each user u_j calculates the shared key $K = z_{j-1}^{n\alpha_j} x_j^{n-1} x_{j+1}^{n-2} \cdots x_{j-2}$. It can be shown that the shared key is actually the quantity $K = g^{\alpha_1\alpha_2 + \alpha_2\alpha_3 + \cdots \alpha_n\alpha_1}$.

In [5], the GDH.1, GDH.2 and GDH.3 protocols are described that extend the two-party DH scheme to the n-party case. The distinguishing characteristic of the GDH.1/2 protocols is that they consist of two stages: an upflow and a downflow stage. For example, in the upflow stage of protocol GDH.1 user u_j receives a message of the form $\{g^{\alpha_1}, g^{\alpha_1\alpha_2}, \cdots, g^{\alpha_1 \cdots \alpha_{j-1}}\}$ and computes $g^{\alpha_1\alpha_2\cdots\alpha_j}$ by taking the last element of the received message and raising it to the α_j power. User u_j then sends to user u_{j+1} the message $\{g^{\alpha_1}, g^{\alpha_1\alpha_2}, \cdots, g^{\alpha_1 \cdots \alpha_{j-1}}, g^{\alpha_1 \cdots \alpha_j}\}$. During the downflow stage, user u_n takes the output of the upflow stage, treats $g^{\alpha_1 \cdots \alpha_n}$ as the key, calculates g^{α_n} and raises the first $n - 2$ elements of the output of the upflow stage to the α_n power. Then user u_n sends user u_{n-1} a message of the form

$\{g^{\alpha_n}, g^{\alpha_1\alpha_n}, \cdots, g^{\alpha_1\cdots\alpha_{n-2}\alpha_n}\}$. User u_j performs likewise, calculating the key $g^{\alpha_1\cdots\alpha_n}$ using the last term of the received message, and forwards to u_{j-1} a message formed by taking the first $j-1$ terms of the received message and raising them to the α_jth power. The GDH.3 scheme is a *centralized* scheme that differs from GDH.1/2 in that one user gathers contributions from all users, performs the majority of the computation for the group, and sends messages to each user that can be used to calculate the group secret. The *centralized* nature of the GDH.3 scheme is a drawback in environments where there is no single entity with significantly more computational capabilities than the others users. Further, an extension to the GDH schemes that incorporates user authentication was presented in [18].

Several measures have been proposed to gauge a conference key protocol's complexity [5,17]. The amount of messages sent and received, as well as the amount of bandwidth consumed are important measures of a protocol's efficiency. Another important measure that arises is the amount of rounds that a protocol takes in order to establish a group secret. A protocol that takes more rounds to establish a shared key is less favorable in environments where time and synchronization are precious resources. In [17], the communication complexity involved in establishing a group key is studied. In this work, lower bounds for the total number of messages exchanged, as well as the amount of rounds needed to establish the group key, were determined. They further present a key establishment scheme based upon a hypercube structure where the amount of rounds needed to establish the key is logarithmic in the group size.

A similar technique was proposed in [21,22], in which the problem of group key establishment was examined in terms of signal flow graphs. The basic approach, called the *butterfly* scheme, had communication flow that was reminiscent of the

butterfly diagrams of FFT calculations. The butterfly scheme used the ING scheme as the basic building block, and provided a broad family of approaches in which the amount of rounds needed to establish the group key is logarithmic in the group size. We revisit the butterfly scheme in the following section.

2.3 Conference Trees and the Butterfly Scheme

The general butterfly scheme is built using the ING scheme. However, since the two-party DH protocol is a special case of the ING scheme, we shall use the two-party DH protocol to introduce the basic ideas involved and then extend to using more general ING schemes. We refer to butterfly schemes built using two-party DH as radix-2 butterfly schemes. The terms *radix* and *butterfly* are borrowed from the signal processing community, and their usage is motivated by the resemblance between the communication flow of our butterfly scheme, and the butterfly signal flow diagrams associated with FFT computations [23]. In our work, the usage of radix refers to the size of the initial subgroups used in the butterfly scheme.

In order to explain the basic idea behind the radix-2 butterfly scheme, suppose that the number of users n is a power of 2. The users are paired up with each other to form two-person subgroups, and a key is established for each of these two-person subgroups using the conventional DH protocol. These subgroups are paired up with each other to form larger 4 member subgroups, and the two-party DH protocol is used to establish a group key for the 4 member subgroups. We may successively group subgroups to form larger subgroups and use two-party DH to ultimately achieve a shared group key.

A formal description of the butterfly scheme for $n = 2^r$ members is as follows. Initially, suppose each user u_j has a random secret integer $\alpha_j \in \mathbf{Z}_p^*$. The n users

are broken into pairs of users $u_j^1 = \{u_{2j-1}, u_{2j}\}$. Here we have used the superscript in the notation to denote which round of pairings we are dealing with, while the subscript references the pair. We also refer to the initial secrets that each user possesses as $x_j^0 = \alpha_j$. In the first round, the members of a pair exchange their calculated $g^{x_j^0}$. For example, u_1 sends $g^{x_1^0}$ to u_2, and u_2 sends $g^{x_2^0}$ to u_1. Then, the users u_{2j-1} and u_{2j} each calculate $x_j^1 = g^{x_1^0 x_1^1} = g^{\alpha_{2j-1}\alpha_{2j}} \pmod{p}$. Since $x_j^1 \in \mathbf{Z}_p^*$, and both members of a pair have established a conventional DH key, we may now group the pairs u_j^1 into a second level of pairs, e.g. $u_1^2 = \{u_1^1, u_2^1\}$, and more generally $u_j^2 = \{u_{2j-1}^1, u_{2j}^1\}$ so that the second level of pairings consists of 4 users in a pair. Each user from u_{2j-1}^1 has an associated member of u_{2j}^1 to whom they send $g^{x_{2j-1}^1}$ and similarly receive $g^{x_{2j}^1}$ from. Every member in u_j^2 can calculate $x_j^2 = g^{x_{2j-1}^1 x_{2j}^1} \pmod{p}$. A third pairing, consisting of 8 users may be formed and a similar procedure carried out if needed. In general, $u_j^k = \{u_{2j-1}^{k-1}, u_{2j}^{k-1}\}$ and $x_j^k = g^{x_{2j-1}^{k-1} x_{2j}^{k-1}} \pmod{p}$. Ultimately, the procedure continues until there are only two intermediate values that can be combined to get the group secret.

A trellis diagram depicting the communication flows between users is depicted in Figure 2.1 (a). It is not necessary that each user perform a communication during each round. In fact, such an operation might use more power since many users are transmitting identical information. In networks, such as wireless networks, where multicasting is available, alternative trellis diagrams can be constructed where one user multicasts an intermediate message to multiple users. An example of such a trellis is depicted in Figure 2.1 (b). An alternative way to view the butterfly scheme is provided in Figure 2.1 (c), which depicts the tree associated with the butterfly scheme. This tree, which we refer to as the *conference tree*, describes the successive subgroups and subgroup keys that are formed en route to establish-

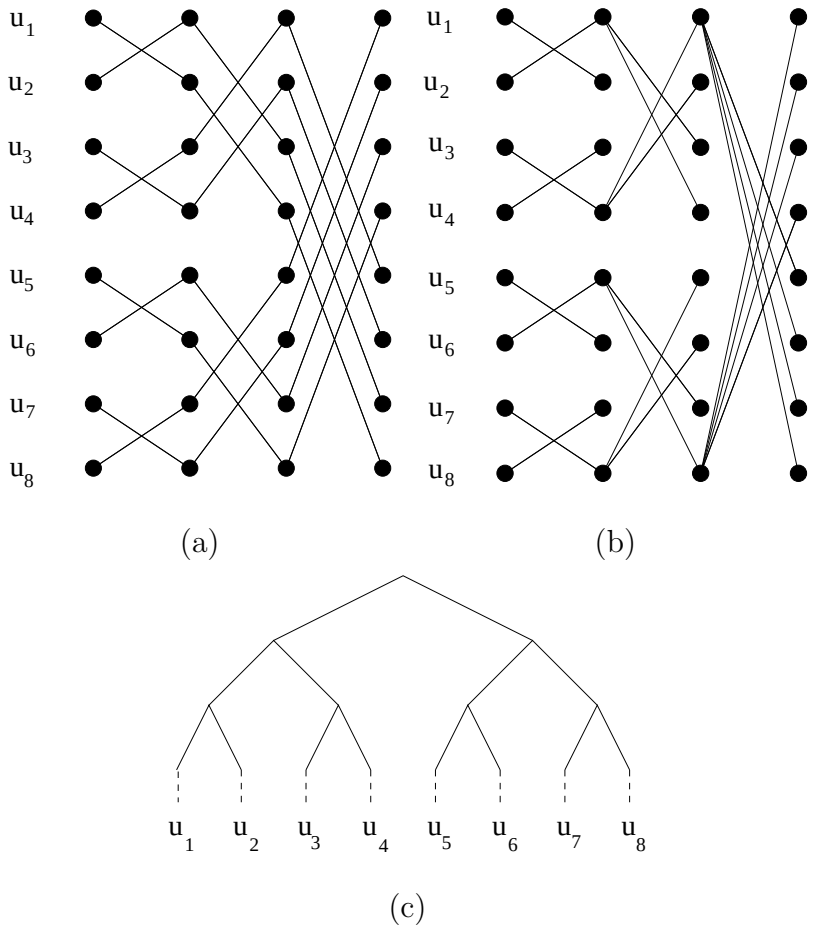


Figure 2.1: The radix-2 butterfly scheme for establishing a group key for 8 users. (a) Without broadcasts, (b) Using broadcasts, and (c) the associated conference tree.

ing the key for the entire group. For example, there is a node on the conference tree that is the grandparent of $\{u_1, u_2, u_3, u_4\}$ and hence there is a subgroup key that can allow $\{u_1, u_2, u_3, u_4\}$ to communicate securely amongst themselves if so desired.

When n is not a power of 2, a group key still can be established easily. In this case, we form a subgroup with an amount of users equal to the largest power of 2 less than or equal to n. The remaining users are further broken down in a similar

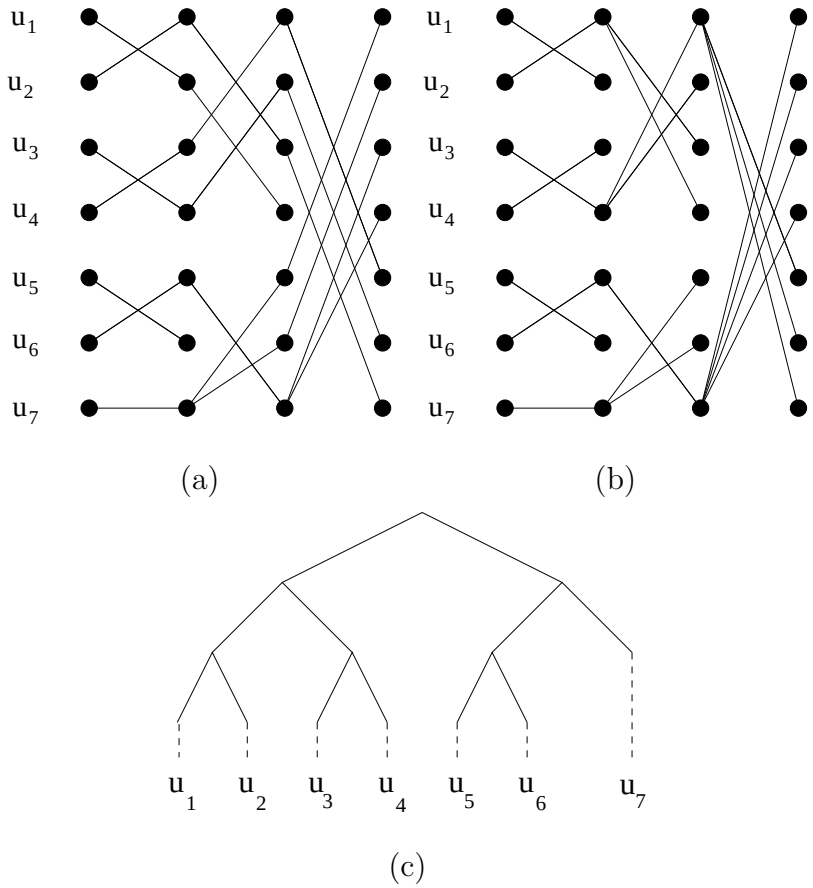


Figure 2.2: The radix-2 butterfly scheme for establishing a group key for 7 users. (a) Without broadcasts, (b) Using broadcasts, and (c) the associated conference tree.

fashion, resulting in a new set of remaining users that can be further broken down. For example, a group of 7 users will be broken down into subgroups of 4, 2, and 1 members. Subgroup and group keys are formed in a fashion similar to the case when n is a power of 2. The trellis and conference tree for $n = 7$ users is depicted in Figure 2.2. The number of rounds needed to complete the radix-2 butterfly scheme is $\lceil \log_2 n \rceil$.

We now extend the approach used above to employ the more general ING scheme as the basic building block. Since the resulting schemes are not built using

for *Stage* $k = 1 : r$ **do**

  Form subgroups $u_j^k = \{u_{p_k(j-1)+1}^{k-1}, u_{p_k(j-1)+2}^{k-1}, \cdots, u_{p_kj}^{k-1}\}$;

  Establish a secret x_j^k for subgroup u_j^k using x_j^{k-1} as secrets in a p_k-member ING scheme ;

end

Algorithm 1: Algorithm for calculating the group key using ING scheme when the group size is factored as $n = p_1 p_2 \cdots p_r$.

a two-party protocol, they are termed non-radix-2 butterfly schemes. Suppose that $n = p_1 p_2 \cdots p_r$ is the number of users, and the p_j are not necessarily prime. The general ING butterfly scheme starts by breaking the group into subgroups of size p_1 and uses the ING scheme to establish a shared key for each of the $n_2 = p_2 \cdots p_r$ subgroups. The n_2 subgroups are further broken down into subgroups consisting of p_2 subgroups, and the ING protocol is used to establish subgroup keys for these larger subgroups. The process continues until a key is established for the entire group. The procedure for this scheme is presented in Algorithm 1, where $u_j^0 = u_j$ and the initial user secrets are $x_j^0 = \alpha_j$. An example is depicted for the case of $n = 9$ users in Figure 2.3. The total amount of rounds is $2 \log_3 9 = 4$, and the amount of messages is 36. The direct use of the ING scheme for 9 users requires 8 rounds and 72 messages. The divide and conquer strategy in the butterfly approach improves the efficiency of the ING scheme. Additionally, the logarithmic amount of rounds needed by the butterfly scheme to establish the group key is an improvement over the linear amount of rounds required by the GDH schemes of [5]. We further note that the hypercube approach of [17] also requires a logarithmic amount of rounds to establish the conference key. However, the hypercube approach does not address the issue of using a general subgroup size as the building block for designing a

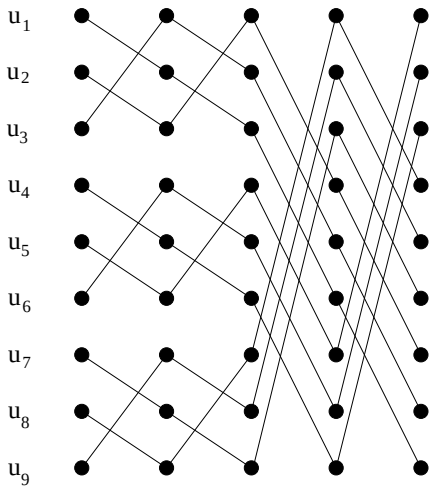


Figure 2.3: The trellis for $n = 9$ users using two levels of 3-party ING scheme.

scalable conference key establishment scheme. By using the ING scheme as the basic module in the butterfly scheme, we include the hypercube approach as a special case, and have generalized their approach. Further, the butterfly scheme described above allows for the use of multicast channels to improve communication efficiency.

It is not necessary to use a factorization of n in designing the non-radix-2 butterfly scheme. In fact, for prime n, this factorization would necessitate using an n-party ING scheme, and require a large amount of rounds in forming the group key. Rather, what is required is that the degrees p_j of the ING schemes used satisfy $\prod p_j \geq n$. In this case, some positions are left unused. For example, when $n = 8$ and $p_1 = p_2 = 3$ one position of a 3-party ING scheme is empty, in which case that computation simply uses the 2-party DH scheme instead

The total number of rounds needed in the ING butterfly scheme for $n = p_1 p_2 \cdots p_r$ users is

$$TR = \sum_{j=1}^{r} (p_j - 1) = \left(\sum_{j=1}^{r} p_j \right) - r. \tag{2.1}$$

When choosing a factorization to represent n, the more factored representation leads to a smaller number of rounds TR. We now show that using a binary conference tree produces the group key in the fewest amount of rounds. To do this, we show that if one uses a p_j ING scheme for round j of the group key establishment, then the use of several two-party DH schemes in place of the p_j ING scheme either produces the same amount of rounds or fewer in establishing the group key.

If we require that all of the computation on one level of a conference tree is completed prior to the formation of the keys in the next level up the conference tree, then using the two-party DH scheme as the building block leads to trees with the least amount of rounds needed to establish the group key. The proof for this claim is provided in the following lemma. Since using two-party DH leads to binary trees that require the least amount of time rounds, we shall restrict our attention to binary trees for the remainder of the chapter.

Lemma 1. *Let n be the amount of users, and suppose that we wish to establish a conference tree where level j uses a p_j ING scheme as the basis, then a binary tree (where $p_j = 2$) produces an optimal conference tree.*

Proof. Suppose that you have an optimal set of numbers $\{p_1, \cdots, p_r\}$ that are used to construct the conference tree for n users. Then the number $N = \prod_{j=1}^{r} p_j \geq n$, and the total rounds $TR = \left(\sum_{j=1}^{r} p_j\right) - r$ is minimal.

We will show that if there is a $p_j \neq 2$ then we may replace p_j by a sequence of numbers all of which have value 2. Suppose there is a j such that $p_j \neq 2$, then the p_j contributes $\Delta_j = p_j - 1$ to the total amount of rounds TR. Define $p'_j = \{2, 2, \cdots, 2\}$ which is a sequence of length $\lceil \log_2 p_j \rceil$. If we use this set of numbers in place of p_j, we instead contribute $\Delta'_j = \lceil \log_2 p_j \rceil$ to the total cost. It is clear that using p'_j in place of p_j produces an $N' \geq n$. However, the incremental

cost $\Delta'_j = \lceil \log_2 p_j \rceil$ is less than or equal to Δ_j (in fact, if $p_j = 3$ then equality holds, else it is strictly less). Thus, if $p_j > 3$ then replacing p_j by p'_j produces a set of numbers with lesser amount of total rounds TR, which contradicts optimality. On the otherhand, if $p_j = 3$ then replacing p_j by p'_j will produce a set of numbers with an equal amount of total rounds TR, and hence we may choose to use p'_j instead of p_j in the construction of the optimal tree. By applying this argument to all $p_j \neq 2$ we conclude that a binary tree must produce an optimal tree. □

It should be pointed out, however, that the argument used above does not produce the optimal tree, but rather only implies that the optimal tree is binary. For example, consider $n = 27$. The total amount of rounds using three levels of 3-party ING is $TR = 6$. If we use the above technique, we replace each 3 by $2 \cdot 2$, and get a conference tree with 2^6 terminal nodes and total cost of 6. However, the optimal tree in this case is the binary tree of depth $\lceil \log_2 n \rceil$, with total rounds $TR = 5$.

In the butterfly schemes described above, the conference trees were almost balanced and full. For example, the conference tree for $n = 8$ users involves 3 levels of internal nodes, and all 8 users are placed at the same depth in the tree. For more arbitrary amounts of users, the users are all roughly placed at the same depth. More general depth assignments and conference tree structures may be given to the users. In the next section, we shall exploit the extra freedom provided by more general binary conference trees by placing users at different depths in order to reduce the total group cost needed to form the group key.

2.4 Computational Considerations

In many application environments the users will have varying amounts of computational resources available. Low-power devices, such as wireless appliances, cannot be expected to expend the same amount of computational effort as high-power devices, such as personal computers, when establishing a group secret. It is therefore important to study the problem of efficiently establishing a conference key while considering the varying user costs.

To accomplish the efficient establishment of a conference key in a heterogeneous environment, we introduce a new entity, called the Conference Keying Assistant (CKA). The CKA is responsible for collecting the users' costs or budgets, determining the appropriate conference keying tree, and conveying the conference tree to the conference members if it is feasible to establish the group key. The CKA is not responsible for performing any computation beyond the calculation of the appropriate conference tree, and therefore only needs to be a *semi-trusted* entity who will accurately convey the conference tree to the conference members. We note that the CKA may be a member of the conference, in which case his duties as CKA are in addition to his role as a group member.

In this section, we present methods that the CKA can employ to design the conference tree that is used by the group members to establish the group secret. In particular, we study two problems: minimizing the total cost in establishing a group key, and the feasibility of establishing the group key in the presence of budget constraints. We present algorithms to efficiently determine the conference keys for each of these problems separately, and then together.

2.4.1 Minimizing Total Cost

First, assume that we have n users, and that each user u_j has a cost $w_j \geq 0$ associated with performing one two-party Diffie-Hellman protocol. For example, this cost might be related to the amount of battery power consumed. Suppose we place the n users on a conference tree with n terminal nodes in such a manner that each user u_j has a length l_j from his terminal node to the root of the conference tree. Our goal is to minimize the total cost $C = \sum w_j l_j$ of this tree.

We first address the question of what is the minimum amount of total computation necessary for establishing the group key for n users. This problem can be addressed using coding theory. If we define p_j as $p_j = w_j / (\sum_k w_k)$, then $\sum_j p_j l_j$ is just a scaling of $\sum_j w_j l_j$ by $W = \sum_k w_k$. Let us define X to be a random variable with a probability mass function given by p_j, then minimizing $\sum_j p_j l_j$ is equivalent to finding a code for X with lengths l_j that minimizes the average code length. We thus infer the following lower bound on the total cost for establishing a group key, which follows from the lower bound for expected codelength of an instantaneous binary code for X [24]:

Lemma 2. *Suppose that n users wish to establish a group secret and each user u_j has a cost w_j associated with performing one two-party Diffie-Hellman protocol. Then the total cost C of establishing the group secret satisfies $-W \sum_j p_j \log_2 p_j \leq C$ where $p_j = w_j / W$.*

The observation that efficiently establishing a group key is related to coding allows the CKA to use procedures from coding theory to determine desirable conference trees. In particular, Huffman coding [25–28] is computationally efficient and yields the optimal conference tree that minimizes the total weighted cost. That is, if C^* is the cost of forming the group key using the Huffman tree, then

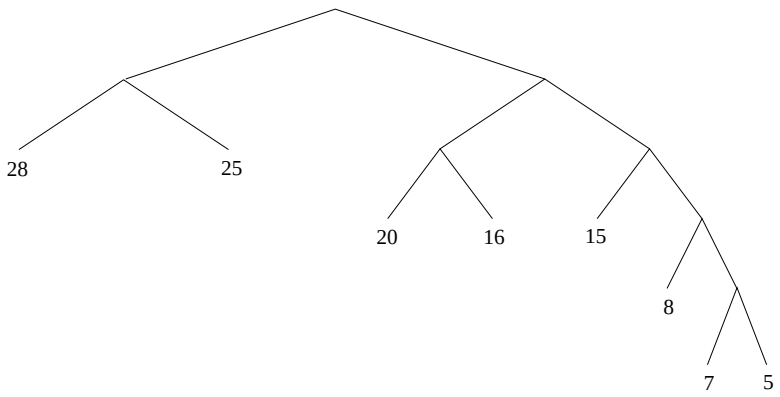


Figure 2.4: Huffman example

the cost C' of using a different conference tree assignment will satisfy $C' \geq C^*$. Since Huffman coding produces an optimal code, we know that the expected cost $\sum_j w_j l_j^*$ satisfies the following bound

$$WH(p) \leq \sum_j w_j l_j^* < W\left(H(p) + 1\right), \tag{2.2}$$

where $H(p)$ is the entropy of the distribution p. Thus, the Huffman construction of the conference key tree has a total cost that is within W of the lower bound.

The following example demonstrates the advantage of using the Huffman algorithm for forming the conference tree when compared to using the full balanced tree of the radix-2 butterfly scheme.

Example 1. *Consider a group of* 8 *users with costs* $w_1 = 28$, $w_2 = 25$, $w_3 = 20$, $w_4 = 16$, $w_5 = 15$, $w_6 = 8$, $w_7 = 7$, *and* $w_8 = 5$. *The Huffman algorithm yields the tree depicted in Figure 2.4. The corresponding length vector is* $l^* = (2, 2, 3, 3, 3, 4, 5, 5)$, *and the total cost is* 351. *The total cost for a full balanced tree is* 372.

We now quantify the improvement that is available when using the Huffman code compared to the cost of using an arbitrary conference tree. For an arbitrary

conference tree, we suppose that the length assigned to user u_j is l_j. If we define a probability distribution q by $q_j = 2^{-l_j}$, then the expected length under the probability p_j of the code with lengths l_j satisfies [24]

$$H(p) + D(p\|q) \leq \sum_{j=1}^{n} p_j l_j < H(p) + D(p\|q) + 1. \tag{2.3}$$

Here $D(p\|q)$ is the Kullback-Leibler divergence between the two probability distributions p and q. The cost for using this tree is $C = W \sum p_j l_j$. We can combine the bound of Equation (2.3) with the bound for the cost of the optimal code $C^* < W(H(p) + 1)$ to get $C - C^* > W\,(D(p\|q) - 1)$. When $D(p\|q) > 1$, this bound is an improvement over the trivial bound $C - C^* \geq 0$.

2.4.2 Budget Constraints

In many cases, the parties wishing to establish a conference key might have a limited budget to spend. The optimal conference tree assignment that results from Huffman coding might assign more computation to some users than they are capable of performing, while assigning less computation to other users than they are capable of performing. In these cases, rather than minimize the total cost, one should ensure that one can first establish the group key, and then reduce the total amount of computation as a secondary issue.

Suppose that user u_j publishes a budget b_j that describes the amount of two-party Diffie-Hellman key establishment protocols he is willing to participate in when establishing the group key. Without loss of generality, we assume that the users' budgets b_j satisfy $b_j \leq b_k$ for $j < k$. We define the budget vector as $b = (b_1, b_2, \cdots, b_n)$. The length vector $l = (l_1, l_2, \cdots, l_n)$ describes the lengths from each user's node to the root of the conference tree. The necessary conditions on

the budget vector b for the existence of a conference key tree with lengths $l_j \leq b_j$ is provided by the Kraft Inequality [24]:

Lemma 3. *Suppose that the budget vector $b = (b_1, b_2, \cdots, b_n)$. Then a conference key tree with lengths l_j exists that satisfies the budget constraint $l_j \leq b_j$ for all j if $\sum_{j=1}^{n} 2^{-b_j} \leq 1$.*

A budget vector that satisfies the Kraft Inequality is said to be *feasible*. When a budget assignment does not satisfy the Kraft Inequality and we choose to drop a single member to generate a feasible budget vector for the remaining users, then the best strategy is to drop the member with the lowest b_1.

Using the budget vector as the length vector does not always lead to a full conference tree in which every node has two children. In order to get a full tree, we must trim the budget vector to produce a length vector l that achieves the Kraft Equality. The length vector is formed by reducing elements of the budget vector by amounts that do not violate the Kraft Inequality. The following lemma provides a useful approach to trimming the length vector assignment while still satisfying the Kraft Inequality.

Lemma 4. *Suppose $b = (b_1, b_2, \cdots, b_n)$ with $b_j \leq b_k$ for $j < k$ satisfies the strict Kraft Inequality, $\sum 2^{-b_j} < 1$, then the modified budget vector c defined by $c = (b_1, b_2, \cdots, b_{n-1}, b_n - 1)$ satisfies the Kraft Inequality $\sum 2^{-c_j} \leq 1$.*

Proof. Observe that 2^{b_n} is the least common denominator of the set 2^{-b_j}. Thus $\sum 2^{-b_j}$ can be expressed as

$$\sum 2^{-b_j} = \frac{x_1 + x_2 + \cdots + x_n}{2^{b_n}} < 1 \tag{2.4}$$

where $x_j = 2^{b_j - b_n}$. In particular, $x_1 + x_2 + \cdots + x_n < 2^{b_n}$, and as a consequence

Data : A length vector l satisfying $\sum 2^{-l_j} \leq 1$.
while $\sum 2^{-l_j} < 1$ **do**
  $j = \arg\max\{l_k\}$;
  $l_j = l_j - 1$;
end

Algorithm 2: Algorithm for calculating the optimal length vector l.

$x_1 + x_2 + \cdots + (x_n + 1) \leq 2^{b_n}$. However, $(x_n + 1)/2^{b_n} = 1/2^{b_n - 1}$, and so the sequence $(b_1, b_2, \cdots, b_n - 1)$ satisfies the Kraft Inequality. □

A consequence of this lemma is that if we subtract 1 from one of the b_j, then choosing the largest b_j least affects $\sum_j 2^{-b_j}$. Using this idea, Algorithm 2 starts with an admissible budget vector b, initializes the length vector $l = b$, and produces a length assignment $l = (l_1, l_2, \cdots, l_n)$ satisfying $l_j \leq b_j$ such that $\sum_j 2^{-l_j} = 1$ and $\sum_j l_j$ is minimized over all length vectors c satisfying $\sum_j 2^{-c_j} \leq 1$. The optimality of this algorithm is discussed in Lemma 5.

As an example of the algorithm, suppose $n = 8$ and that the initial budget is $b = (1, 3, 3, 4, 5, 5, 6, 8)$. This budget vector is feasible and performing the algorithm gives the final assignment $l = (1, 3, 3, 4, 4, 4, 5, 5)$.

Lemma 5. *Algorithm 2 produces an optimal length assignment vector l to the problem*

$$\left\{ \min_l \sum_j l_j \ : \ 1 \leq l_j \leq b_j, \sum_j 2^{-l_j} \leq 1, l_j \in \mathbf{Z}^+ \right\}. \tag{2.5}$$

Proof. We will aim to show that there is an optimal solution in which one decreases the largest value of the budget vector by one. Let l^* be an optimal solution to the problem. Then by the previous lemma $\sum 2^{-l_j^*} = 1$. Consider a sequence of steps that take the budget vector b to the optimal length vector l^* by decreasing one

element by 1 during each step. We denote by J_* the sequence of indices involved in going from b to l^*, where $J_*(k)$ refers to the index of the budget vector that is decreased during kth step. Let j_0 be the index of the largest element of b, we claim there is an optimal solution l' with a corresponding $J_*(1) = j_0$. If $J_*(1) = j_0$ then we are done. However, if $J_*(1) \neq j_0$ then there are two cases. The first case is that there is another element of J_* that has value j_0, in which case we may switch that element with $J_*(1)$ to produce a new sequence of steps that does not alter the value of $\sum 2^{-l_j^*}$ and maintains the optimality of $\sum_j l_j$. The second case is that $j_0 \notin J_*$. If there are any other elements of b with the same value as b_{j_0}, then indices of these may be used in place of j_0, and considered in the preceding argument. However, if there are no b_j's with the same value as b_{j_0} then we seek a contradiction as to the optimality of l^*. Choose an arbitrary element of J_*. This element, which we denote by $J_*(k)$, by assumption has the property that it $b_{J_*(k)} < b_{j_0}$. Define $J_*^- = \{J_*(1), \cdots, J_*(k-1), J_*(k+1), \cdots\}$, which corresponds to the sequence of steps involved in J_* excluding the kth step. Define $J_\sharp = j_0 \| J_{*-}$, which describes a new sequence of steps that starts with j_0 and then the steps of J_{*-}. Then $J_\sharp$ leads to a length vector $l^\sharp = l^* + e_{J_*(k)} - e_{j_0}$, where e_j is the vector of all zeros except in the jth index which has value 1. This length vector has the property that $\sum 2^{-l_j^\sharp} < \sum 2^{-l_j^*}$ since $2^{-l_{j_0}^*} < 2^{-l_{J_*(k)}^*}$. Hence $\sum 2^{-l_j^\sharp} < 1$. However, by the preceding lemma, this means that $l^\sharp$ can be used to produce a better length vector, which contradicts the optimality of l^*.

Hence, the optimal solution may as well have the first step reduce the largest element of the budget vector. Now the problem reduces to finding an optimal solution to the new budget $b' = b - e_{j_0}$. By induction on the number of steps, we therefore conclude that choosing the largest element during each step yields an

optimal solution, and hence the greediness of Algorithm 2 is optimal. □

2.4.3 Combined Budget and Cost Optimization

We have studied the problem of minimizing the total cost of establishing a group key using a tree structure, and whether a group key can be established in a budget-limited scenario. We now address the more realistic scenario where users have different costs as well as budget constraints. We are therefore interested in the problem of minimizing the total cost of the length assignments l_j for the weights w_j given the budget constraint $l_j \leq b_j$. This problem is formally stated as:

$$\begin{aligned} \text{Minimize} \quad & \sum_{j=1}^{n} w_j l_j \\ \text{subject to} \quad & 1 \leq l_j \leq b_j, \quad \sum_{j=1}^{n} 2^{-l_j} = 1, \quad l_j \in \mathbf{Z}^+ \end{aligned}$$

where $\mathbf{Z}^+$ denotes the non-negative integers. Once a length vector has been determined, it can be sorted in ascending order to describe a conference tree.

This problem is more difficult than either the minimum cost problem or the budget-constrained problem. If the budget vector is constant, i.e. $b_j = b$ for every j, then the methods of length-constrained source codes may be applied [29–35]. One efficient algorithm for finding the optimal Huffman code under the maximum codeword length constraint is presented in [29], which is based on the algorithm of [31]. A near optimal solution can be found using Lagrange relaxation, and an efficient implementation is described in [32]. However, in the more general case where the budgets vary from user to user, it is difficult to find the optimal solution since the ordering $w_j \leq w_k$ does not imply $l_j \geq l_k$.

Two suboptimal approaches that employ a greedy strategy were developed to tackle the general problem where the budgets vary from user to user. The first

Data : A budget vector b.
if $\sum 2^{-b_j} > 1$ **then**
  No solution. ;
end
$l = b$;
while $\sum 2^{-l_j} < 1$ **do**
  Let $\delta = 1 - \sum 2^{-l_j}$;
  $K = -\lceil \log_2 \delta \rceil$;
  $J = \{j : l_j \geq K\}$;
  Let $j_0 = \arg\max_{j \in J} w_j$;
  $l_{j_0} = l_{j_0} - 1$;
end

Algorithm 3: Algorithm for calculating the length vector l, given budget b and costs w_j.

algorithm, described in Algorithm 3, is a variant of Algorithm 2, which starts with a length assignment $l = b$ and chooses to decrease the element l_j of the length vector that most reduces the total cost $\sum w_k l_k$ at that step while maintaining the Kraft Inequality. This greedy algorithm is not optimal, as can be seen by the example $b = (2, 2, 3, 3)$ with costs $w = (10, 7, 6, 6)$. In this example, the algorithm produces the length vector $l = (1, 2, 3, 3)$ (which has a total cost of 60), whereas the optimal length vector is $l^* = (2, 2, 2, 2)$ (which has a total cost of 58).

Algorithm 3 is a naive greedy algorithm. By slightly altering this algorithm, another greedy algorithm may be developed with better performance. Instead of decreasing the element that best decreases the total cost, Algorithm 4 chooses to decrease the element with the largest value $w_j 2^{l_j}$. This corresponds to choosing the element that would have the largest change in the cost function per change in the

Data : A budget vector b.
if $\sum 2^{-b_j} > 1$ **then**
  No solution. ;
end
$l = b$;
while $\sum 2^{-l_j} < 1$ **do**
  Let $\delta = 1 - \sum 2^{-l_j}$;
  $K = -\lceil \log_2 \delta \rceil$;
  $J = \{j : l_j \geq K\}$;
  Let $j_0 = \arg\max_{j \in J} w_j 2^{l_j}$;
  $l_{j_0} = l_{j_0} - 1$;
end

Algorithm 4: Improved algorithm for calculating the length vector l, given budget b and costs w_j.

Kraft Inequality. A similar strategy is often used in designing incremental resource allocation schemes in operations research [36]. Algorithm 4 is also suboptimal, but exhibits better performance than Algorithm 3 with a negligible increase in the amount of computation needed. The optimal solution to the combined budget and cost optimization problem can be obtained by performing either full-search, or using the methods of integer programming [37–39]. One useful approach is to apply the branch and bound method to the problem [37, 40–42].

We now compare the near-optimal results of Algorithm 4 with the optimal solution. We performed a simulation where each user's budget b_j was chosen uniformly from $[1, 3n]$, and weights w_j were chosen uniformly from $[1, 100]$. The optimal solution, l^*, was compared with the approximate solution, $\hat{l}$, from Algorithm 4 via

Group size n	$\bar{\rho}$
5	0.0037
6	0.0046
7	0.0027
8	0.0020
9	0.0025
10	0.0020
11	0.0016

Table 2.1: Comparison between the optimal solution and the approximate solution of Algorithm 4 for different group sizes n.

the relative difference

$$\rho = \frac{C(\hat{l}) - C(l^*)}{C(l^*)}. \tag{2.6}$$

This quantity was calculated and averaged over 100 realizations to produce the mean relative difference $\bar{\rho}$, for the group sizes $n = 5$, 6, 7, 8, 9, 10, and 11. The results are presented in Table 2.1, which indicates that Algorithm 4 produces the group key with cost that is within 0.5% of the optimal cost. Due to the computational complexity required to find the optimal solution for 100 realizations, we only present results through $n = 11$.

Since determining the optimal solution is very computationally intensive for large group sizes, it is unreasonable for the CKA to find the optimal conference tree when users have both budget constraints and varying costs. Instead, Algorithm 4, although not optimal, has very competitive performance and its computational requirements are small compared to full-search or the branch and bound method, and is a reasonable candidate for the CKA to use in determining the conference

tree.

2.5 Efficiency and Feasibility Evaluation

We now compare our tree-based conference key establishment schemes with other schemes in the literature. We assume that no broadcast channels are available, and that if one user desires to communicate amongst many, he must establish many separate connections. There are two evaluations that we present: first, we consider the total cost needed to establish a group key when the users have different costs; second, we examine the feasibility of establishing a conference key when group members have different budget constraints.

2.5.1 Comparison of Total Cost

We simulated a scenario in which there were three classes of users. The first class corresponds to users with a large amount of computational power (and hence lower user cost), the second corresponds to a medium level of computational power, and the last class represents users with low-powered devices or a high cost. In order to represent this distinction, the users were assumed to have weights drawn according to three different distributions. For every 10 users, 2 users have weights drawn according to the first distribution, 5 according to the second distribution, and 3 according to the third distribution. The first weight distribution was a discrete uniform distribution with integer values from $[1, 50]$, while the second was a discrete uniform distribution over $[501, 550]$, and the third was a discrete uniform distribution over $[951, 1000]$.

We compared the total cost for the Huffman scheme with the cost of the butter-

fly scheme, the ING scheme, the GDH.1/2 scheme, and the GDH.3 scheme. Since there are differences between the communication and computational procedures of the different schemes, we assume that the user costs are associated with the cost to perform the two modular exponentiations needed in a two-party DH scheme. This means, for example, that if a user has a cost of w to perform one round of two-party DH, then he has a cost of $3w/2$ to perform a 3-party ING scheme since there are 3 modular exponentiations involved.

We also assume that every user in a DH scheme performs the two modular exponentiations. For example, if the subgroup $\{u_1, u_2\}$ share a secret x and the subgroup $\{u_3, u_4\}$ share a secret y, and use DH to establish a shared key for the 4 members, then both u_1 and u_2 calculate g^x and g^{yx}. Similarly, both u_3 and u_4 calculate g^y and g^{xy}. In actuality, however, only one member from each subgroup must calculate and transmit the message g^x or g^y. The costs for the Huffman and butterfly schemes that we report do not reflect this possible savings, and are therefore overestimates of the actual costs.

The total cost required to establish the conference key was calculated for different group sizes and averaged over 500 realizations. The average costs are depicted in Figure 2.5. Examining Figure 2.5 we see that the ING and GDH.1/2 schemes have higher total cost than the Huffman, butterfly, and GDH.3 schemes. In this example, the Huffman scheme performs better than the butterfly scheme by an average of 6.7%. GDH.3 has the best performance in terms of total cost. However, GDH.3 is a *centralized* scheme and cannot be categorized as a completely *distributed* conference keying scheme since one user performs the majority of the computations for the group. In contrast, the Huffman scheme and the butterfly scheme are contributory and do not make any single user responsible for the ma-

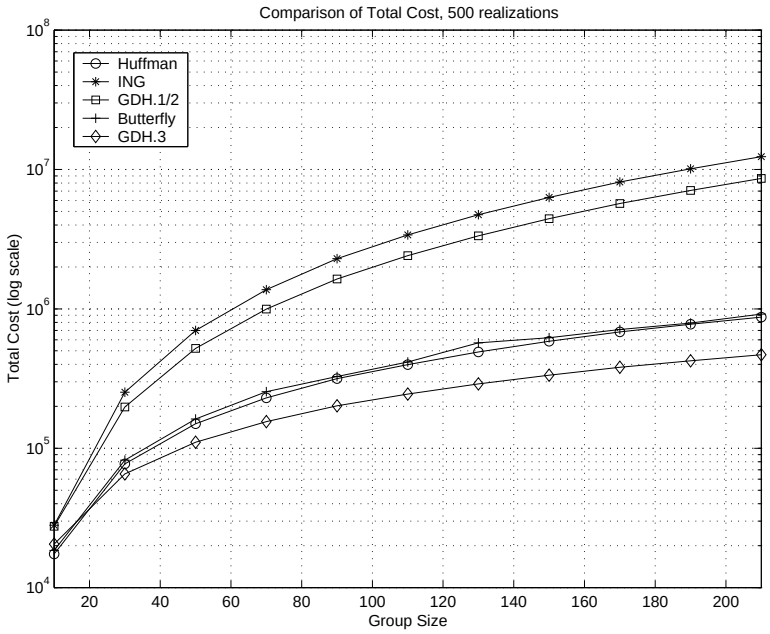


Figure 2.5: Cost comparison of establishing a conference key using the Huffman-based conference tree, the ING scheme, GDH.1/2, the butterfly scheme, and the GDH.3 scheme. The first four schemes are contributory protocols, while GDH.3 is a centralized protocol.

jority of the computation (although they do allot more load to some users than others). In scenarios where it is appropriate to have one user or entity do nearly all of the work for the remaining users the use of centralized multicast key distribution schemes [6, 8, 14] will lead to more efficient distribution of keying information than conference keying schemes.

2.5.2 Feasibility Comparison

Another major concern is the feasibility of establishing a secure conference in the presence of budget constraints. When the users have different budgets, it might not be possible for different schemes to establish a conference key. We shall quantify

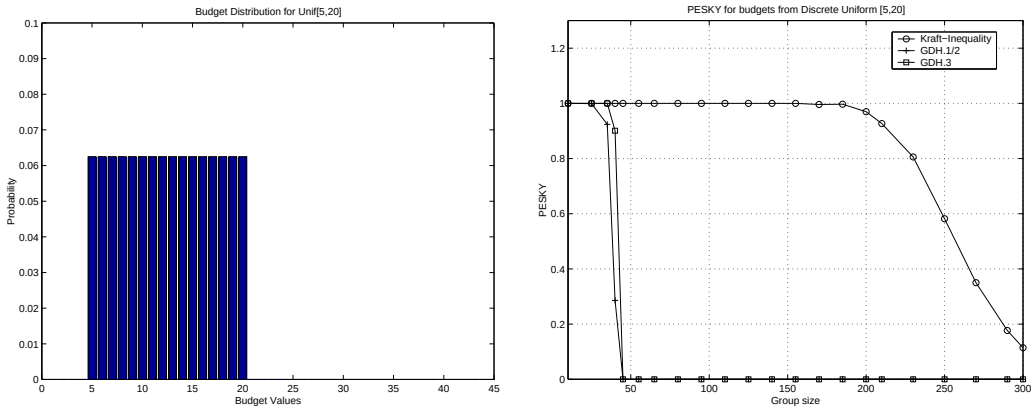
the likelihood that a conference key can be established in a scenario where the users' budgets are drawn according to a distribution by introducing the PESKY (Probability of Establishing the Session KeY) measure.

Suppose that $\mathcal{B}$ denotes the set of all possible budget vectors for n users, and that μ is a probability distribution over $\mathcal{B}$ describing the likelihood of the users having a certain budget vector. Let a conference key scheme be denoted by F, and $F(\mathcal{B})$ the set of all budget vectors $\mathcal{B}$ which are feasible for F. Then formally, the PESKY measure is defined as:

$$PESKY(F, n) = \sum_{b \in F(\mathcal{B})} \mu(b). \tag{2.7}$$

For example, if we let F refer to a conference tree scheme built using Algorithm 2, Algorithm 3, or Algorithm 4, then a budget vector is feasible if it satisfies the Kraft Inequality, and therefore $F(B) = \{b : \sum_j 2^{-b_j} \leq 1\}$. In general, it is difficult to find closed form expressions for PESKY, and Monte Carlo methods must be used to estimate PESKY.

We used PESKY to study the likelihood that different schemes could produce a group key when the user's budgets were drawn according to different distributions. We assumed that the budgets b_j correspond to the amount of two-party DH schemes that a user is willing to participate in, and that the two modular exponentiations are the most significant expense for the user. Therefore, each value of the budget allows for 2 modular exponentiations to be performed. As before, we assume that every user in a subgroup performs both of the modular exponentiations in a DH scheme. We compare the PESKY for Algorithms 2-4 with PESKY for both the GDH.1/2 and GDH.3 schemes for three different budget distributions. The PESKY for Algorithms 2-4 are conservative estimates of the actual probability of establishing the session key since it is not necessary that all members of a



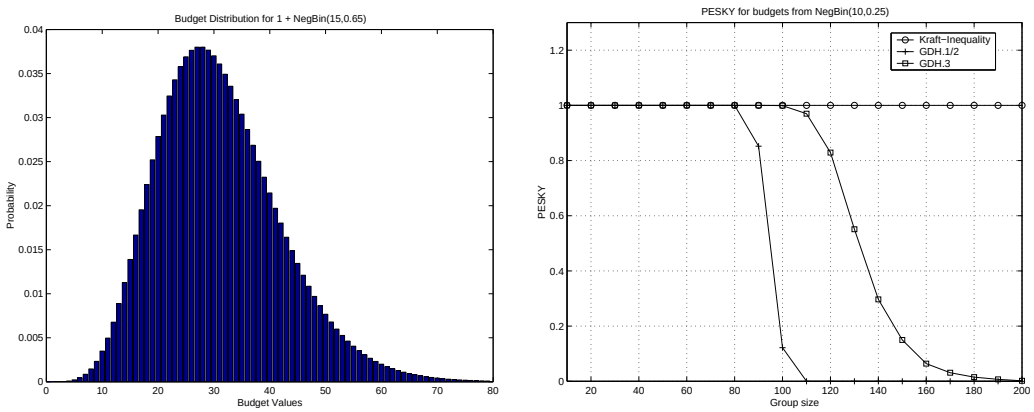
(a) (b)

Figure 2.6: (a) Budget distribution discrete uniform with integer values from $[5, 20]$ (b) Corresponding PESKY

subgroup perform the first modular exponentiation of a DH scheme.

The first budget distribution is a discrete uniform distribution with integer values from $[5, 20]$. The distribution is presented in Figure 2.6(a), and the corresponding PESKY curves are presented in Figure 2.6(b). Since the GDH schemes require that one user performs an amount of modular exponentiations equal to the amount of users n, it is impossible for groups of more than 40 users to be formed via the GDH protocols with this distribution, as can be seen in Figure 2.6(b). The PESKY plots for this distribution demonstrate that it is more likely that a budget vector can satisfy the Kraft Inequality than the requirements of either the GDH.1/2 or GDH.3 schemes. In fact, it is not until the group sizes become larger than $n = 200$ that a significant decrease is observed in the likelihood of forming a group key using a conference tree.

For the second distribution, the elements of the budget vector are generated as $1 + NegBin(10, 0.25)$, where $NegBin(s, p)$ is the negative binomial distribution



(a) (b)

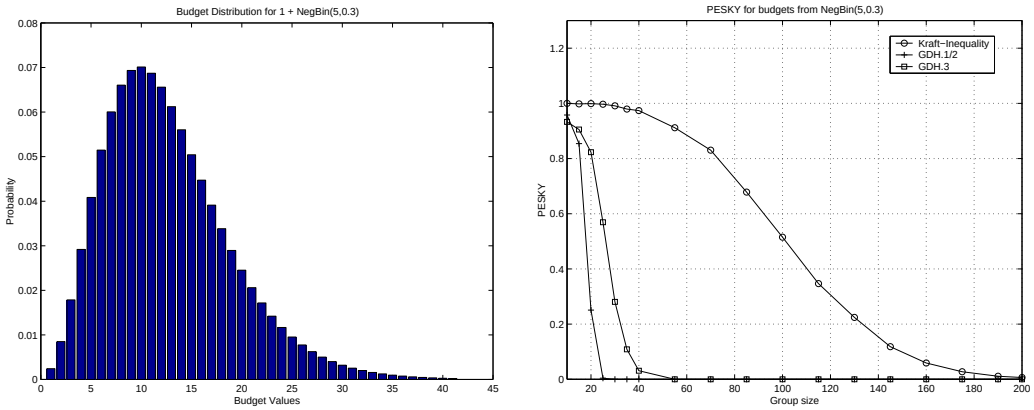
Figure 2.7: (a) Budget distribution, shifted version of a negative binomial distribution with parameters $s = 10$, and $p = 0.25$. (b) Corresponding PESKY

with probability mass function $q(b)$ given by:

$$q(b) = \binom{s+b-1}{b} p^s (1-p)^b \quad \text{for } b \in \{0, 1, \cdots\}. \tag{2.8}$$

The addition of 1 to $NegBin(10, 0.25)$ was to ensure that no users had a budget of 0. In Figure 2.7, we see that the tree-based schemes exhibit a 100% likelihood of successfully establishing the conference key for conferences with between 10 and 200 users. The PESKY values for GDH.3 begins to drop off at $n = 100$ users while the values for GDH.1/2 drop off at $n = 80$ users. Since a large amount of budget values are above 20, the PESKY curves do not drop off as quickly as they did for the uniform case.

In the third distribution, the budgets were drawn according to $1+NegBin(5, 0.3)$, as depicted in Figure 2.8 (a), and the corresponding PESKY measures are depicted in Figure 2.8 (b). This distribution describes a similar phenomenon to the uniform distribution above, but includes a heavier tail at higher budget values that could represent a diminishing class of more powerful users. The fact that roughly 6%



(a) (b)

Figure 2.8: (a) Budget distribution, shifted version of a negative binomial distribution with parameters $s = 5$, and $p = 0.3$. (b) Corresponding PESKY

of this distribution corresponds to budget values below 5 has a significant effect upon the PESKY plots. In particular, we see that the PESKY all of the conference keying schemes drop off earlier than in Figure 2.6 (b). For example, when there are $n = 70$ users there is only an 80% chance of forming a conference key using one of these schemes with this distribution compared to a 100% chance with the distribution of Figure 2.6 (a). We also see that the GDH.1/2 schemes are very unlikely to successfully establish a group key, even for group sizes of $n = 20$, and that all of the GDH schemes are unable to establish a group key for groups of more then 60 users.

Therefore, in resource-limited scenarios, the choice of which conference keying scheme is very critical. The GDH.3 scheme, although cost-efficient, obtains this efficiency at the expense of requiring a single user have significantly more power and resources than the other users. In applications where the users have a more balanced distribution of resources, the GDH schemes have PESKY graphs that rapidly drop off and are therefore unlikely to successfully establish a group key. In

these cases, the conservative estimates of PESKY for tree-based conference keying schemes indicate that they are more likely to establish a group key, and Algorithm 4 is a judicious choice for constructing the conference tree since it requires little computational effort and has near-optimal performance.

2.6 System Sensitivity to False Costs

In this section, we examine the effect that announcing costs different from the true user costs has upon the total cost of using the Huffman conference tree. There are two cases that we consider. First, we consider the issue that users announce costs that are approximations of the true costs. Next, we examine the case where some of the users are untrusted, and announce large costs for the purpose of reducing their individual cost. We present an approach that controls the detrimental effect that greedy users have upon the total cost.

2.6.1 Sensitivity to Approximate Costs

We begin by considering that the true user costs are $\hat{w}_j \in [1, \hat{B}]$, where $\hat{B}$ is a suitable upper bound placed on the exact costs. We suppose the costs that the users announce are derived by applying an operator T to $\hat{w}_j$, i.e. $w_j = T(\hat{w}_j)$. We define $\hat{W} = \sum_j \hat{w}_j$, and $\hat{p}_j = \hat{w}_j / \hat{W}$. If we build a code using p_j with lengths l_j, then the average length under $\hat{p}$ is $\sum_j \hat{p}_j l_j$. We show that if we design the code to minimize $\sum p_j l_j$, then we can design the operator T such that $\sum |\hat{p}_j l_j - p_j l_j|$ is small. Since $l_j \leq n$, we get $\sum_{j=1}^{n} |\hat{p}_j l_j - p_j l_j| \leq n \left(\sum_{j=1}^{n} |\hat{p}_j - p_j| \right)$. We now derive a bound for $\sum_{j=1}^{n} |\hat{p}_j - p_j|$:

$$\sum_{j=1}^{n} |\hat{p}_j - p_j| \quad = \quad \sum_{j=1}^{n} \left| \frac{w_j}{W} - \frac{\hat{w}_j}{\hat{W}} \right| \tag{2.9}$$

$$\leq \frac{1}{W\hat{W}}\left[\sum_{j=1}^{n}|w_j(\hat{W}-W)|+\sum_{j=1}^{n}|W(w_j-\hat{w}_j)|\right] \tag{2.10}$$

$$\leq \frac{1}{W\hat{W}}\left[2W\sum_{j=1}^{n}|w_j-\hat{w}_j|\right] \tag{2.11}$$

$$= \frac{2}{\hat{W}}\left[\sum_{j=1}^{n}|w_j-\hat{w}_j|\right]. \tag{2.12}$$

We consider two cases for the operator T. The first case we consider is when T is a clipping operator, namely

$$T_{\hat{B}}(\hat{w}) = \begin{cases} \hat{w} & : \quad \hat{w} \leq \hat{B} \\ \hat{B} & : \quad \hat{w} > \hat{B} \end{cases}.$$

It is clear that as $\hat{B} \to \infty$, we have more $w_j = T_{\hat{B}}(\hat{w}_j) = \hat{w}_j$, and thus the bound (2.12) tends to 0 as we increase $\hat{B}$. We shall examine the clipping operator later in this section. The second operation we consider is quantization. Here we consider the interval $[1, \hat{B}]$ divided into N equally sized quantization bins. The operator T then maps $\hat{w}$ to the nearest quantization value, and $|w_j - \hat{w}_j| \leq \hat{B}/(2N)$. In this case, we get

$$\sum_{j=1}^{n}|\hat{p}_j - p_j| \leq \frac{1}{\hat{W}}\left(\frac{\hat{B}n}{N}\right) \tag{2.13}$$

which tends to 0 as the number of quantization bins N increases. Therefore, in both the case of clipping and quantization, the parameters can be adjusted to bring the probability distribution p close to $\hat{p}$, and thus the designed average codelength $\sum p_j l_j$ close to the average codelength of using l_j under $\hat{p}$.

2.6.2 Sensitivity to Costs from Untrusty Users

We next consider the effect one user has upon the computational cost of the remaining users. In many scenarios, there may be a user that hurts the other users

by either selfishly trying to make his cost small, or maliciously trying to make the total cost of the remaining users large. Recall that if the weights are ordered as $w_1 \geq w_2 \geq \cdots \geq w_n$ then the lengths of the Huffman code can be ordered as $l_1^* \leq l_2^* \leq \cdots \leq l_n^*$ [24]. Therefore, if a user would like to keep his cost as small as possible, he should announce as large of a weight as possible. Additionally, announcing a large weight causes the p_j of the other users to decrease, thereby increasing their codelengths (see [43] for the relationship between a symbol's codelength and its self-information). Thus, if a malicious user wishes to adversely affect the lengths of the other users, he should announce as large of a weight as possible.

We first derive the worst-case effect that one user can have upon the computational cost of the other group members when Huffman coding is used to construct the conference tree. We suppose that the malicious or selfish user is u_1, and that he publishes a large weight w_1. To determine how much extra cost does choosing a large w_1 impose upon the other $n-1$ users, we define $\check{W} = \sum_{k=2}^{n} w_k$ and define the probability $q_j = w_j/\check{W}$ for $j \in \{2, 3, \cdots, n\}$, and $q_1 = 0$. Then q_j represents the probabilities that would be used in constructing a conference tree if user u_1 were not participating. Let l_j^* denote the optimal codelengths constructed using p_j, and $\check{l}_j^*$ be the optimal codelengths constructed using q_j. Since u_1 is not involved in the construction of $\check{l}_j^*$, we have $\check{l}_1^* = 0$.

We define the following quantities:

$$C^* = \sum_{j=1}^{n} w_j l_j^*, \quad \check{C}^* = \sum_{j=2}^{n} w_j \check{l}_j^*, \quad C_{ex}^* = \sum_{j=2}^{n} w_j l_j^*.$$

We are interested in comparing C_{ex}^*, which is the total cost of the remaining $n-1$ users given the probabilities p_j which incorporate u_1's cost, with $\check{C}^*$, which is the total cost of the $n-1$ users $u_2, u_3, \cdots, u_n$ without considering u_1's announced cost.

Since $\check{C}^*$ arises as the optimal code for the $n-1$ users with costs $w_2, w_3, \cdots, w_n$,

we know $\check{C}^*$ minimizes costs of the form $\sum_{j=2}^{n} w_j l_j$. In particular, C_{ex}^* must satisfy:

$$C_{ex}^* = \sum_{j=2}^{n} w_j l_j^* \geq \sum_{j=2}^{n} w_j \check{l}_j^* = \check{C}^*. \tag{2.14}$$

We may derive an upper bound for C_{ex}^* by observing that the code given by $\check{l}_j^*$ can be used to construct a code for p_j by taking $l_1 = 1$ and $l_j = \check{l}_j^* + 1$. The optimal code for the weights $w_1, w_2, \cdots, w_n$ must be better than this code, and hence

$$C^* \leq w_1 + \sum_{j=2}^{n} w_j(\check{l}_j^* + 1) = \check{C}^* + W. \tag{2.15}$$

Since $C_{ex}^* = C^* - w_1 l_1^*$, we have $C_{ex}^* \leq \check{C}^* + W - w_1 l_1^* \leq \check{C}^* + \check{W}$. Gathering the results together, we get the overall bound $\check{C}^* \leq C_{ex}^* \leq \check{C}^* + \check{W}$. The upper bound is achieved when $w_1 > \check{W}$, and hence, in the worst case, u_1 forces the other $n-1$ users to spend an extra $\check{W}$ of resources.

Next, we consider the more general case where a fraction of the users are untrusty and announce large costs. Suppose that the true costs are $\hat{w}_j$, and that the announced costs are $\tilde{w}_j$. If the underlying statistics governing $\hat{w}_j$ are known, it is possible to determine which $\tilde{w}_j$ are outliers and remove those users from the group key formation procedure. However, in many cases, the value of the conference exists regardless of whether a few users were untrusty, and it is desirable to have those users in the conference. In this case, an approach must be used to reduce the detrimental effect of these bad users upon the cost of forming the entire group key.

We suppose that the CKA applies a clipping operator to the announced user costs $\tilde{w}_j$ to produce costs $w_j = T_B(\tilde{w}_j)$ that are used by the CKA in determining the conference tree. Ideally, we would like to build the conference tree using the exact costs $\hat{w}_j$, but these are not available. Instead, if the conference tree is built

using w_j or $\tilde{w}_j$, the corresponding lengths l_j and $\tilde{l}_j$ are used with the exact costs $\hat{w}_j$, which can lead to an increase in the total cost.

To study the amount of additional cost incurred by using a code designed for w_j when the true costs are $\hat{w}_j$, we shall examine the average codelength. Hence we design codes for $p_j = w_j/W$ and $\tilde{p}_j = \tilde{w}_j/\tilde{W}$, where $\tilde{W} = \sum \tilde{w}_j$. We are interested in studying $\sum \hat{p}_j l_j$ and $\sum \hat{p}_j \tilde{l}_j$. The Kullback-Leibler divergence $D(\hat{p}\|p)$ describes the additional average codelength that different coding schemes incur when designed for the wrong distribution p when the correct distribution is $\hat{p}$ [24, 34, 43, 44]. Given a model distribution for the true user costs, the CKA can use $D(\hat{p}\|p)$ to determine the value of the clipping parameter B that minimizes the miscoding penalty.

We calculated the divergence for $n = 100$ users when the original costs w_j were drawn according to $10LN(0, 1) + 100$, where $LN(\mu, \sigma)$ is the lognormal distribution arising from a normal distribution with mean μ and variance σ. The lognormal distribution was chosen because it has a long tail. The probability that a user is untrusty was 0.05, and untrusty users were assumed to announce a cost $\tilde{w}_j = \hat{w}_j + Y$, where $Y = 1000$ and $w_j = T_B(\tilde{w}_j)$. The choice of $Y = 1000$ was arbitrary and chosen to represent a large bias that an untrusty user might place on his announced costs. An example divergence $D(\hat{p}\|p)$ for costs $\hat{w}_j$ drawn according to this distribution is presented in Figure 2.9. There is a minimum that appears at approximately $B^* = 150$. A system should be designed for the average case. In order to do this, the optimal clipping parameter should be averaged over many realizations of the costs. For costs drawn according to $10LN(0, 1) + 100$, we averaged the optimal clipping value over 10000 realizations and found the mean optimal clipping value to be $\overline{B}^* = 150.44$ and the variance of the optimal clipping

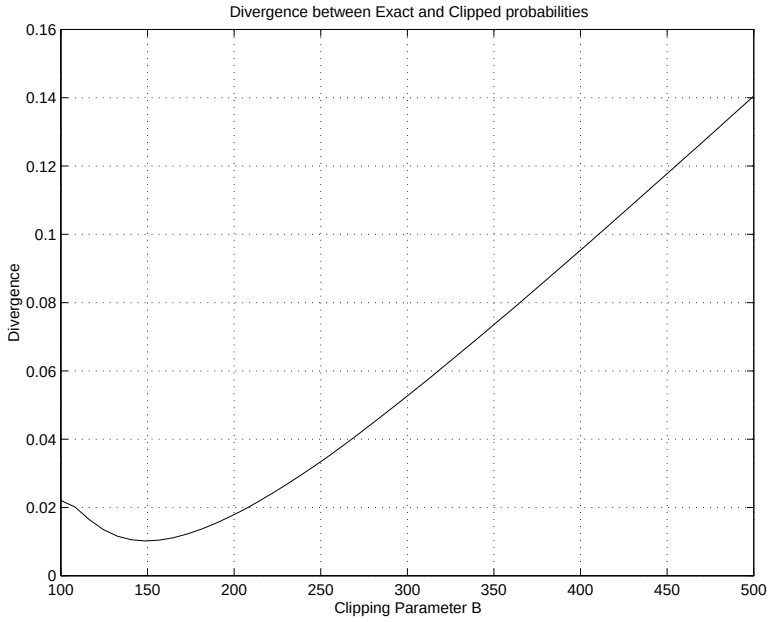


Figure 2.9: An example divergence $D(\hat{p}\|p)$ where $\hat{w}_j \sim 10LN(0,1) + 100$, and $w_j = T_B(\hat{w}_j)$.

value as $\sigma_{B^*} = 25.60$.

The relative difference between the cost of using the Huffman-based conference tree using w_j and $\hat{w}_j$ are now compared. If $\hat{l}_j$ are the optimal codelengths using $\hat{w}_j$, $\tilde{l}_j$ are the optimal codelengths constructed using $\tilde{w}_j$, and l_j are the optimal codelengths constructed using w_j, then we are interested in comparing

$$\rho = \frac{\sum_j \hat{w}_j l_j - \hat{w}_j \hat{l}_j}{\sum_j \hat{w}_j \hat{l}_j} \quad \text{and} \quad \tilde{\rho} = \frac{\sum_j \hat{w}_j l_j - \hat{w}_j \tilde{l}_j}{\sum_j \hat{w}_j \hat{l}_j}.$$

We calculated these values for the case when the exact costs were drawn according to $10LN(0,1) + 100$ with $Y = 1000$, while the probability of a user being untrusty was 0.05. The results were averaged over 100 realizations and are presented in Figure 2.10. The quantity ρ is presented for different clipping parameter values, and we observe that there is a range of minimal values from $B = 140$ to $B = 220$, which is roughly the region that the divergence curves predict. The clipped relative

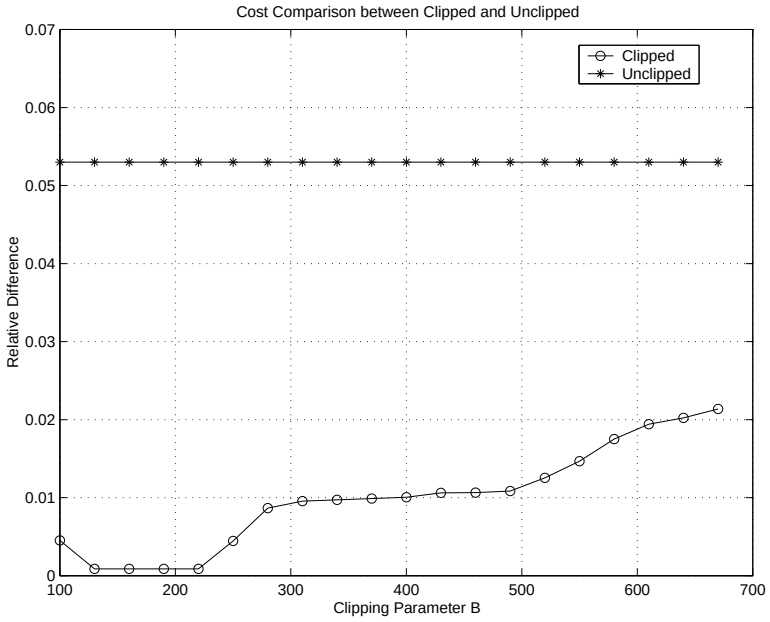


Figure 2.10: The relative costs ρ and $\tilde{\rho}$ are presented for when the exact user costs are drawn as $\hat{w}_j \sim 10LN(0,1) + 100$, and that there is an 0.05 likelihood that a user is untrusty, and $Y = 1000$.

costs show a significant improvement over the unclipped relative costs. Without performing the clipping, the untrusty users force the entire group to spend an average of over 5% more than if the exact user costs were used. By performing the clipping operation, however, this detrimental effect can be significantly lessened to less than 0.5%.

2.7 Chapter Summary

In this chapter we have presented methods for establishing a conference key that are based upon the design of an underlying tree called the conference tree. In heterogeneous environments, where users have varying costs and budgets, the conference tree can be designed to address the user differences. We examined the

design of the conference tree for three different cases. First, we studied the problem of minimizing the total cost of establishing the group key when the users had different costs. The problem of designing the conference tree was related to source coding, and techniques for designing source codes, such as Huffman coding, were employed to design the conference tree. The second case we investigated was when the users had the same cost, but different budget requirements. We observed that a necessary condition for a conference tree to exist for a given vector of budget requirements is that the budget vector satisfies the Kraft Inequality. We then presented a greedy algorithm that trimmed a feasible budget vector to achieve a length assignment that optimally reduces the total length of the conference tree. Finally, the third case we examined is when the users have both varying costs and budget requirements. We presented a computationally efficient near-optimal algorithm using a greedy incremental resource assignment strategy that achieves a total cost within 0.5% of the optimal solution for small group sizes.

We presented simulations comparing the total cost of the butterfly and Huffman-based schemes against the scheme of Ingemarsson et al., and the GDH family proposed by Steiner et al. Out of the class of non-centralized conference keying schemes, the Huffman scheme exhibited the least total cost. In situations where no single user has an extremely large budget, centralized conference keying schemes are unlikely to successfully establish a conference key. To investigate this phenomenon, we introduced the PESKY measure, which describes the probability that a conference keying scheme can establish a session key in the presence of budget constraints. We provided simulations where the user budgets were drawn according to different distributions, and in all cases the PESKY values for different group sizes were higher for our tree-based schemes than for either the GDH.1/2 or

the GDH.3 schemes.

Next, we examined the effect that using false user costs would have on the total cost. It was shown that by increasing the quantization resolution, or by increasing the threshold level, that the difference between the total cost of using the exact and approximate costs for a given length assignment tends to 0. We then examined the effect a subset of users who falsely announce large costs has upon the total cost. In order to reduce the detrimental effect of designing a conference tree for falsely announced user costs, we proposed the use of a clipping operator to prevent untrusty users from being too greedy and minimize the divergence to determine the optimal threshold value. Simulations using the Huffman algorithm to construct the conference tree show that the optimal threshold values agree with those predicted by the divergence.

In the next two chapters we will explore issues related to the maintenance of key information for group applications where users are allowed to join and depart the service. Whereas this chapter has focused on protocols for contributory key agreement, the next two chapters will employ a centralized entity for the management of key information.

Chapter 3

Key Management and Distribution for Secure Multimedia Multicast

3.1 Introduction

Several key technologies have matured in the past decade, allowing for the possibility of building new infrastructures for the delivery and consumption of multimedia content. The combination of well-developed multimedia standards, such as MPEG-4 and H.324, and advances in both wireless and networking technologies is creating opportunities for new commercial markets such as HDTV, wireless video, and pay-per-view services [45–48]. Integral to many of these ventures is the ability to broadcast or multicast identical data simultaneously to groups of users. Multicast communications is efficient since it reduces the demands on network and bandwidth resources. It will play a key role in delivering services shared by many users, such as pay-per-view broadcasts of sporting events, as well as allowing for interactive multimedia applications such as interactive television, video conferencing, and communal gaming. However, before such group-oriented commercial ventures can be successfully deployed, the issue of controlling access to multimedia content

must be addressed. Service providers must be able to ensure the availability of multimedia data to privileged (paying) members while preventing unauthorized access to this data by non-privileged users.

The most appropriate framework for securing server-oriented content distribution is by using a centralized entity that is responsible for maintaining the integrity of the users' keys. The problem of maintaining access control is difficult when the content is being distributed to a group of users since the membership will most likely be dynamic with users joining and leaving the service. Unlike unicast communication, the departure of a group member does not imply the termination of the communication link. In addition, upon departing the service, users must be de-registered and prevented from obtaining future multicasts. Similarly, when new members join the service, it is desirable to prevent them from accessing past content.

The problem of key management for multicasts has seen recent attention in the literature, and several efficient schemes have been proposed with desirable communication properties [6–8,49]. These schemes, however, do not consider application-specific properties that might affect the design of an access control system. In particular, multimedia data has rich properties, such as the capability to have information invisibly hidden in it [12, 13, 50, 51] and operate in a scalable or layered format [45], which we can exploit to achieve an improved design of an access control system for multimedia multicasts.

In this chapter we study the problem of key management for multimedia multicast services. We begin by discussing the fundamental problem of distributing secret information to a group of privileged users, and present information theoretic derivations associated with multicast key management. Next, in Section 3.3, we

introduce multicast key management and present a basic key management scheme that will be used later in the chapter in Section 3.4. In Section 3.5, we introduce two different modes for distributing the rekeying messages associated with securing multimedia group communication. The first, and more conventional, approach employs the use of a media-independent, or external channel, to convey the rekeying messages. A second mode, a media-dependent channel, is achieved for multimedia by using data embedding techniques. We explore the advantages and disadvantages of these different techniques. In Section 3.6, we introduce a new message format for multicast key management that uses one-way functions to securely distribute new key material to subgroups of users. An advantage of this approach over the traditional message format is that no additional messages must be sent to flag the users which portion of the message is intended for them, thereby reducing communication overhead. We then show how to map the message to a tree structure in order to achieve desirable scalability in communication and computational overhead. Next, in Section 3.7, we study the interplay between the key management scheme and the mode of conveyance by studying the feasibility of embedding rekeying messages using a data embedding method that has been recently proposed for fractional-pel video coding standards such as H.263 and MPEG-2. In Section 3.8, we extend the key management scheme to multilayer multimedia applications where group members may subscribe or cancel membership to some layers while maintaining membership to other layers. Finally, we present some conclusions in Section 3.9.

3.2 Basic Multicast Information Theory

This section describes a summary of information theory results relevant to multicasting. This summary is based upon results that were presented in [9] and [52], but have been rederived and put into a context relevant to the work proposed.

First, let $\mathcal{U} = \{u_1, u_2, \cdots, u_n\}$ denote the user set consisting of n users u_j. Associated with each user u_j is a private key K_j that is drawn uniformly from a key space $\mathcal{K}$. Of the n users, only a subset of them will be privileged. We denote the set of private keys associated with privileged members by K_P, and the set of private keys associated with non-privileged users by K_F. For example, if there are $n = 4$ users, and users u_1, u_3 are privileged, then $K_P = \{K_1, K_3\}$, and $K_F = \{K_2, K_4\}$. There is a secret S that is drawn from a space $\mathcal{S}$ that the group center wishes to transmit to members of the multicast group $\mathcal{U}$. The broadcast message α is a function of the secret S, as well as the private user information of the privileged users, $\alpha = f(S, K_P)$.

It is useful to derive bounds on the size of the broadcast message given the following security constraints:

- The user's secrets K_P and the secret S uniquely determine the broadcast message

$$H(\alpha|S, K_P) = 0. \tag{3.1}$$

- Knowing *only* a user's private key K_j does not decrease the uncertainty of the secret S. That is

$$H(S|K_j) = H(S). \tag{3.2}$$

  In particular, this implies that $H(S|K_P) = H(S)$.

- No uncertainty in the secret remains if both a user's private key K_j and the broadcast message are available.

$$H(S|K_j, \alpha) = 0. \tag{3.3}$$

- The broadcast message does not decrease the uncertainty in a user's private key:

$$H(K_j|\alpha) = H(K_j). \tag{3.4}$$

- The broadcast message alone does not decrease the uncertainty in the secret:

$$H(S|\alpha) = H(S). \tag{3.5}$$

The first results that we present are from Just et al. [9].

Lemma 6. *The entropy of the broadcast message α is equal to mutual information between the message and the joint random variable (K_P, S):*

$$H(\alpha) = I(\alpha; K_P, S). \tag{3.6}$$

Proof. We start by applying the chain rule to the mutual information:

$$\begin{aligned} I(\alpha; K_P, S) &= I(\alpha; K_{j_1}) + I(\alpha; K_{j_2}|K_{j_1}) + \cdots \\ &+ I(\alpha; K_{j_m}|K_{j_1}, K_{j_2}, \cdots, K_{j_{m-1}}) + I(\alpha; S|K_P). \end{aligned}$$

Expanding the mutual information terms yields the telescoping sum:

$$\begin{aligned} I(\alpha; K_P, S) &= H(\alpha) - H(\alpha|K_{j_1}) + H(\alpha|K_{j_1}) - H(\alpha|K_{j_1}, K_{j_2}) + \cdots \\ &+ H(\alpha|K_P) - H(\alpha|K_P, S), \end{aligned}$$

which yields

$$I(\alpha; K_P, S) = H(\alpha) - H(\alpha|K_P, S). \tag{3.7}$$

However, $H(\alpha|K_P, S) = 0$, so that

$$I(\alpha; K_P, S) = H(\alpha). \tag{3.8}$$

□

Lemma 7. *Let $D \subset P$ be a subset of privileged members such that $|D| \leq m-1$, and let K_D be the set of private keys associated with users in D. Let K_i be a private key of a user $u_i \in P - D$. Then for a secret S and a broadcast message α, we have*

$$H(K_i) - H(K_i|\alpha, K_D) \geq H(S). \tag{3.9}$$

Proof. The term $H(K_i, S|\alpha, K_D)$ may be expanded in two different ways:

$$\begin{aligned} H(K_i, S|\alpha, K_D) &= H(K_i|\alpha, K_D) + H(S|\alpha, K_D, K_i) && (3.10) \\ &= H(S|\alpha, K_D) + H(K_i|\alpha, K_D, S). && (3.11) \end{aligned}$$

Since $H(S|\alpha, K_j) = 0$ for any user u_j in the privileged set P, we have that $H(S|\alpha, K_D, K_i) = H(S|\alpha, K_D) = 0$, and thus

$$H(K_i|\alpha, K_D) = H(K_i|\alpha, K_D, S). \tag{3.12}$$

Observe that since $I(K_i; S|\alpha) = I(S; K_i|\alpha)$ we have

$$\begin{aligned} H(K_i|\alpha) - H(K_i|\alpha, S) &= H(S|\alpha) - H(S|\alpha, K_i) \\ H(K_i) - H(K_i|\alpha, S) &= H(S). && (3.13) \end{aligned}$$

Since $H(K_i|\alpha, S) \geq H(K_i|\alpha, S, K_D)$, we may apply (3.12) to get $H(K_i|\alpha, S) \geq H(K_i|\alpha, K_D)$. Substituting this result into (3.13) gives $H(K_i) - H(K_i|\alpha, K_D) \geq H(S)$. □

A consequence of this lemma is the fact that each private key K_i will have entropy greater than the entropy of secret, i.e. $H(K_i) \geq H(S)$. We may now put these results together to get a lower bound on the size of the broadcast message given the conditions stated.

Theorem 1. *Suppose that the keys K_j are distributed independently of each other, i.e. $H(K_j, K_i) = H(K_j) + H(K_i)$, and the conditions (3.1)-(3.5) hold, then the following bound on the size of the broadcast message holds:*

$$H(\alpha) \geq mH(S) \tag{3.14}$$

Proof. By Lemma 6 we have

$$\begin{aligned} H(\alpha) &= I(\alpha; K_P, S) && (3.15)\\ &= I(\alpha; K_P) + I(\alpha; S|K_P) && (3.16)\\ &= I(K_P; \alpha) + I(S; \alpha|K_P) && (3.17)\\ &= H(K_P) - H(K_P|\alpha) + H(S) - H(S|\alpha, K_P). && (3.18) \end{aligned}$$

Using the fact that $H(S|\alpha, K_P) = 0$ and that

$$\begin{aligned} H(K_P) &= H(K_{j_1}, K_{j_2}, \cdots, K_{j_m}) && (3.19)\\ &= H(K_{j_1}) + \cdots + H(K_{j_m}), && (3.20) \end{aligned}$$

which follows from the independence of the private keys, we have

$$H(\alpha) = H(K_{j_1}) + \cdots + H(K_{j_m}) - H(K_P|\alpha) + H(S). \tag{3.21}$$

Similarly, expanding $H(K_P|\alpha)$ using the chain rule gives

$$\begin{aligned} H(K_P|\alpha) &= H(K_{j_1}|\alpha) + H(K_{j_2}|\alpha, K_{j_1}) + \cdots \\ &\quad + H(K_{j_m}|\alpha, K_{j_1}, \cdots, K_{j_{m-1}}). && (3.22) \end{aligned}$$

Upon substitution we get

$$H(\alpha) = H(K_{j_1}) - H(K_{j_1}|\alpha) + \sum_{i=2}^{m}\Big(H(K_{j_i}) - H(K_{j_i}|\alpha, K_{j_1}, \cdots, K_{j_{i-1}})\Big) + H(S). \tag{3.23}$$

By observing that $H(K_{j_1}|\alpha) = H(K_{j_1})$, and applying Lemma 7 we get the desired result $H(\alpha) \geq mH(S)$. □

In summary, we have presented two main results from [9] that govern the theoretical underpinnings of multicast key management. The first result that was shown states that the entropy of a user's private information must be greater than the entropy of the secret that is to be distributed to the group. This translates into the security terminology by implying that the bit length of the user's private key should be as large as the bit length of the group secret. It was also shown, under the assumption of independent keys, that the size of the broadcast message must be at least as large the size of the group times the size of the secret that is to be conveyed. This latter result gives a lower bound on the communication requirements for rekeying. In particular, it implies that the best that can be done is a message whose size is linear in the amount of group members unless the key independence assumption is relaxed. Currently, the most popular family of multicast key management schemes are those that employ a tree key hierarchy, in which the key information that each user has is not independent of each other.

3.3 Multicast Key Management Schemes

The distribution of identical data to multiple parties using the conventional point-to-point communication paradigm makes inefficient usage of resources. The redundancy in the copies of the data can be exploited in multicast communication

by forming a group consisting of users who receive similar data, and sending a single message to all group users [53]. Access control to multicast communications is typically provided by encrypting the data using a key that is shared by all legitimate group members. The shared key, known as the session key (SK), will change with time, depending on the dynamics of group membership as well as the desired level of data protection. Since the key must change, the challenge is in key management– the issues related to the administration and distribution of keying material to multicast group members.

In order to update the session key, a party responsible for distributing the keys, called the group center (GC), must securely communicate with the users to distribute new key material. The GC shares keys, known as key encrypting keys (KEKs), that are used solely for the purpose of updating the session key and other KEKs with group members.

As an example of key management, we present a basic example of a multicast key distribution scheme. Suppose that the multicast group consists of n users and that the group center shares a key encrypting key with each user. Upon a member departure, the previous session key is compromised and a new session key must be given to the remaining group members. The GC encrypts the new session key with each user's key encrypting key and sends the result to that user. Thus, there are $n-1$ encryptions that must be performed, and $n-1$ messages that must be sent on the network. The storage requirement for each user is 2 keys while the GC must store $n+1$ keys. This approach to key distribution has linear communication, computation and GC storage complexity. As n becomes large these complexity parameters make this scheme undesirable, and more scalable key management schemes should be used.

In general, during the design of a multicast application, there are several issues that should be kept in consideration when choosing a key distribution scheme. We now provide an overview of some of these issues.

- **Dynamic nature of group membership:** It is important to efficiently handle members joining and leaving as this necessitates changes in the session key and possibly any intermediate keying information.

- **Ability to prevent member collusion:** No subset of the members should be able to collude and acquire future session keys or other member's key encrypting keys.

- **Scalability of the key distribution scheme:** In many applications the size of the group may be very large and possibly on the order of several million users. The required communication, storage, and computational resources should not become a hindrance to providing the service as the group size increases.

In Section 3.2, we summarized the work of [9, 52] for the distribution of secret information via broadcast messages. These results provide a insight into the communication resources needed to achieve the above goals. In particular, it was shown in Theorem 1 that for a key size of B bits, the message needed to update a group of n users must be at least nB bits to provide *perfect security* in the key distribution. One key result of [9] is that in order to achieve a smaller broadcast size, it is necessary to do away with the constraint that the private information held by each user is mutually independent. Therefore, to reduce the usage of communication resources, the users must share secret information.

One strategy for having users share secret information is to arrange the keys

according to a tree structure. The tree based approach to group rekeying was originally presented by Wallner et al. [7], and independently by Wong et al. [6]. In such schemes an a-ary tree of depth $\log_a n$ is used to break the multicast group into hierarchical subgroups. Each member is assigned to a unique leaf of the tree. KEKs are associated with all of the tree nodes, including the root and leaf nodes. A member has knowledge of all KEKs from his leaf to the root node. Thus, some KEKs are shared by multiple users. Adding members to the group amounts to adding more depth to the tree [54], or adding new branches to the tree [6]. Upon member departure the session key and all the internal node KEKs assigned to that member become compromised and must be renewed. Due to the tree structure, the communication overhead is $\mathcal{O}(\log n)$, while the storage for the center is $\mathcal{O}(n)$ and for the receiver is $\mathcal{O}(\log n)$.

Various modifications to the tree scheme have been proposed. In [8], a modification to the scheme of Wallner et al. is presented. By using pseudo-random generators, their scheme reduces the usage of communication resources by a factor of two. Similarly, Balenson et al. [54] were able to reduce the communication requirements by a factor of two using one-way function trees. The security of the Canetti et al. scheme can be rigorously proven, while the security of the approach using one-way function trees is based upon non-standard cryptographic assumptions and has therefor not been rigorously shown. In [49] Canetti et al. examine the tradeoffs between storage and communication requirements, and a modification to the tree-based schemes of [6, 7] is presented that achieves sublinear server-side storage. Further, in [55], it was shown that the optimal key distribution for a group leads to Huffman trees and the average number of keys assigned to a member is related to the entropy of the statistics of the member deletion event.

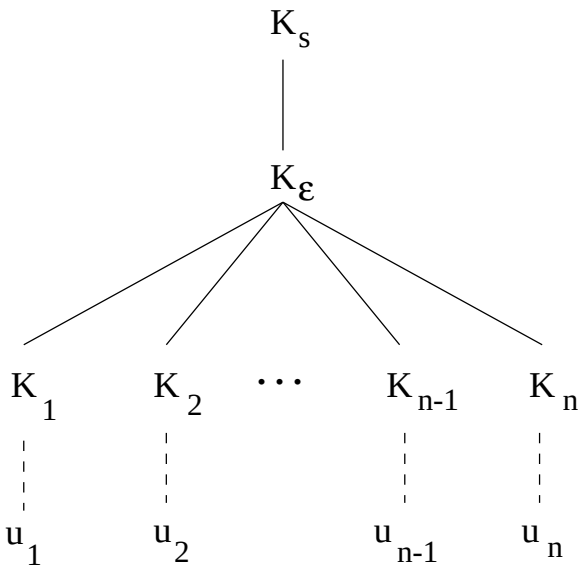


Figure 3.1: The basic key distribution scheme.

3.4 A Basic Key Management Scheme

In this section, we present a simplified key management scheme that will be used in the discussions in Section 3.6.1 where we introduce an improved format for the rekeying message. The key management scheme presented here is an elementary key management scheme that consists of two layers of KEKs, and a SK that is used to protect the bulk content.

Consider a group of n multimedia users who will share a multimedia multicast. In the simple key distribution scheme for n users, depicted in Figure 3.1, user u_j has two key encrypting keys K_j and K_ϵ, and the session key K_s. The KEK K_ϵ is the root KEK and is used to encrypt messages that update K_s. The remaining keys $K_1, K_2, \cdots, K_n$ are KEKs that are used to protect updates of K_ϵ.

Due to the dynamic nature of the group, and the possible expiration of keying material, it is necessary to update both the SK and KEKs using rekeying messages. The three operations involved are key refreshing, key updating when a new user

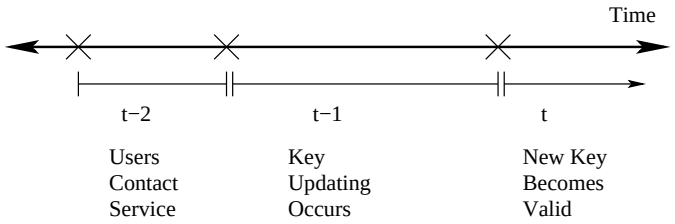


Figure 3.2: The time intervals $t-2$, $t-1$ and t used in the paper. The joining/departing user contacts the service during time interval $t-2$, the rekeying messages are transmitted during $t-1$, and new key information takes effect at the beginning of time interval t.

joins the service, and key updating when a user departs the service. In the the discussions that follow, we use an integer-valued time index to denote the time intervals during which fundamental operations occur, and assume that there is a system-level mechanism that flags or synchronizes the users to the same time frame. We shall always use the time index t to denote the interval for which the new key information will become valid. Time interval $t-1$ will correspond to the time interval during which the new key information is being transmitted. Further, time interval $t-2$ corresponds to the interval of time during which a new member contacts the service provider wishing to join, or a current member announces to the service provider his desire to depart the service. We have depicted these cases in Figure 3.2. Observe that it is not necessary that the time intervals have the same duration.

3.4.1 Key Refreshing

Refreshing the session key is important in secure communication. As a session key is used, more information is released to an adversary, which increases the chance

that a SK will be compromised. Therefore, periodic renewal of the session key is required in order to maintain a desired level of content protection. By renewing keying material in a secure manner, the effects of a session key compromise may be localized to a short period of data.

The cryptoperiod associated with a session key is governed by many application-specific considerations. First, the value of the data should be examined and the allowable amount of unprotected (compromised) data should be addressed. For example, the broadcast of a sporting event might allow the data to be unprotected for a short period, whereas a video conference between corporate executives would likely have stricter security requirements and necessitate more frequent key refreshing.

Since the amount of data encrypted using KEKs is usually much smaller than the amount of data encrypted by a session key, it is not necessary to refresh KEKs. Therefore, KEKs from the previous time interval $t-1$ carry over to the next time interval. In order to update the session key $K_s(t-1)$ to a new session key $K_s(t)$, the group center generates $K_s(t)$ and encrypts it using the root KEK $K_\epsilon(t)$. This produces a rekeying message $\alpha_s(t) = E_{K_\epsilon(t)}(K_s(t))$, where we use $E_K(m)$ to denote the encryption of m using the key K. The message $\alpha_s(t)$ is sent to the users.

3.4.2 Member Join

In multimedia services, such as pay-per-view and video conferences, the group membership will be dynamic. Members may want to join and depart the service. It is important to be able to add new members to any group in a manner that does not allow new members to have access to previous data. In a pay-per-view system, this amounts to ensuring that members can only watch what they pay for,

while in a corporate video conference there might be sensitive material that is not appropriate for new members to know.

Suppose that, during time interval $t-2$, a new user contacts the service desiring to become a group member. If there were $n-1$ users at time $t-2$ then there will be n users at time t. During time interval $t-1$, the rekeying information must be distributed to the $n-1$ current members. Observe that we must renew both the SK and the root KEK in order to prevent the new user from accessing previous rekeying messages and to prevent access to prior content.

The first stage of the key updating procedure requires updating the root KEK from $K_\epsilon(t-1)$ to $K_\epsilon(t)$. Since all of the members at time $t-1$ share $K_\epsilon(t-1)$, the group center may communicate the new KEK $K_\epsilon(t)$ securely to these members by forming and transmitting the message $\alpha_\epsilon(t) = E_{K_\epsilon(t-1)}(K_\epsilon(t))$. Next, the service provider generates a new session key $K_s(t)$ and updates the session key using a rekeying message of the form $\alpha_s(t) = E_{K_\epsilon(t)}(K_s(t))$.

Meanwhile, during time interval $t-1$, the new user completes registration with the service and is given the new keys $K_s(t)$, $K_\epsilon(t)$, and K_{n+1}. This completes the actions required during time interval $t-1$, and at the start of time interval t all of the $n+1$ members have the new keying material.

3.4.3 Member Departure

Let us consider the case when user u_n leaves the group at time frame $t-1$. Since user u_n knows $K_s(t-1)$ and $K_\epsilon(t-1)$ these keys must be renewed. First K_ϵ is renewed. To accomplish this the GC forms a new key $K_\epsilon(t)$ and encrypts it with the keys K_j for $j \neq n$. A single message

$$\alpha_\epsilon(t) = E_{K_1}(K_\epsilon(t)) \| E_{K_2}(K_\epsilon(t)) \| \cdots \| E_{K_{n-1}}(K_\epsilon(t)) \tag{3.24}$$

is formed and sent to all the users using either the media-independent or media-dependent channel. Here we use the symbol $\|$ to denote concatenation of bit streams. Next, the session key is updated. The GC forms a new SK $K_s(t)$ and encrypts using the new KEK $K_\epsilon(t)$ to form $\alpha_s(t) = E_{K_\epsilon(t)}(K_s(t))$. This message is then sent to the users.

As a final note, we observe that the size of this message agrees with Theorem 1. Here, the message α consists of the concatenation of $n-1$ smaller messages. The message α is distributed to the $n-1$ remaining users. The total length of the message is $n-1$ times the block size of the encryption algorithm employed. Since the key length is smaller than the block size, we agree with Theorem 1.

3.5 Distribution of Rekeying Messages for Multimedia

After the formation of the rekeying messages, they must be delivered to the users. This issue is rarely considered in the secure multicast literature. However, it is an integral part of a system's design. For the transmission of multimedia data, we have identified two distinct classes of mechanisms, depicted in Figure 3.3, that are available for the delivery of the rekeying messages:

- **Media-Independent Channel:** In this mode, the rekeying messages are conveyed by a means totally disjoint from the multimedia content.
- **Media-Dependent Channel:** A media-dependent channel exists when the media is capable of having a small amount of data imperceptibly hidden inside the host media.

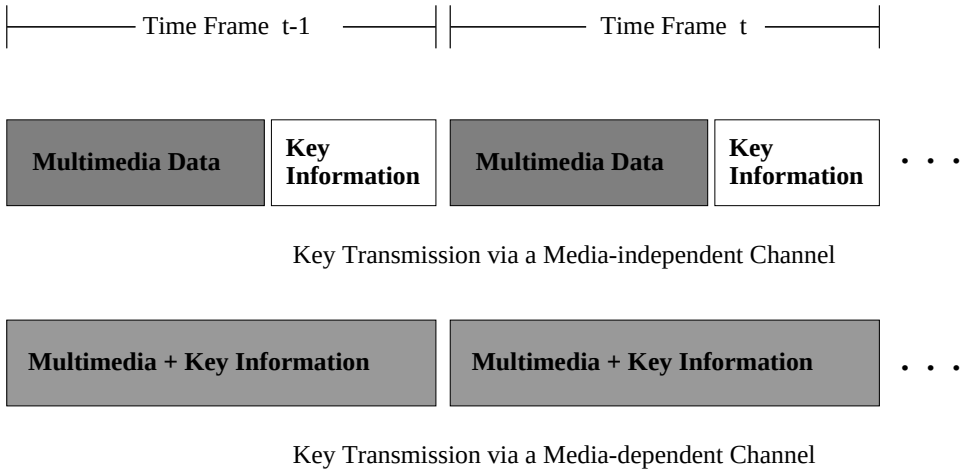


Figure 3.3: Two approaches to distributing the key information in multimedia multicasting: (a) using a media-independent channel, and (b) using a media-dependent channel.

In a conventional, non-secure multimedia application, the multimedia data consists of multiple streams. Depending on the application, these streams may either be multiplexed together and placed onto the network, or treated as separate layers that are passed onto a separate delivery protocol. For example, in MPEG-2 Systems a multiplexer operation will multiplex the audio and video data into either a program stream or a transport stream [45]. As another example, MPEG-4 provides packetized elementary streams to the Delivery Multimedia Integration Framework (DMIF) which deals with different delivery scenarios and allows for desirable delivery techniques such as unequal error protection (UEP) [56, 57].

The location of the encryption operation in a multimedia application's design, as well as the mode that the encryption operates under, has a significant effect on the performance of the multimedia multicast service. In Figure 3.4, we present a generic diagram that captures the multiplexing involved in H.324, MPEG-2 Systems program stream or transport stream, or the operations of the

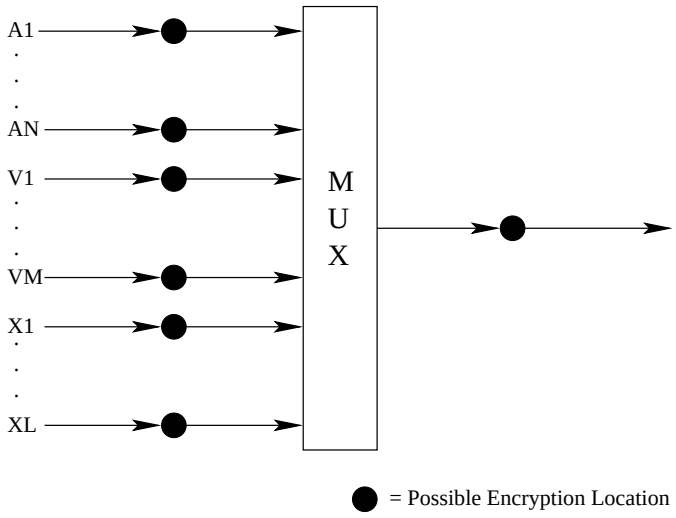


Figure 3.4: A generic multiplexing diagram depicting several audio streams (A1 - AN), video streams (V1 - VM), and auxiliary streams (X1 - XL). Also depicted are locations where encryption is possible.

DMIF in MPEG-4. Several audio streams (A1 - AN), video streams (V1 - VM), and auxiliary streams (X1 - XL) are fed as input into the multiplexer. Upon output is a data stream that consists of packets that have been interlaced. With respect to this diagram, there are two locations where encryption can be placed. Encryption can either be located before the multiplexer or after it. If encryption is placed after the MUX, then there are two manners in which it can encrypt the data stream. First, it can encrypt each packet individually, thereby maintaining the separation of the packets that was introduced by the multiplexer. The second option is for the encryption to operate in a streaming mode, such as cipher block chaining [58], whereby the separation between different media packets will be lost. The disadvantage of the latter mode of operation is that it is no longer possible to treat the layers separately, which is essential to performing important delivery techniques like UEP. Therefore, if encryption is placed after the MUX, it should

maintain the separation between the packets.

However, it is not necessary to place the encryption after the MUX to maintain the separation between the layers. In fact, placing the encryption before the MUX will encrypt each media or object stream independently, and the multiplexer will interleave the various encrypted streams into separated packets. The multiplexer and transmitter will then maintain the separation between the different media streams, which is essential for reliable delivery of multimedia. Therefore, there is no advantage in placing encryption after the MUX since the segregation between the different streams should be maintained, and hence encryption should be done prior to multiplexing. In fact, in the MPEG-4 IPMP framework IPMP control points are located prior to the DMIF at the encoder [59].

For the remainder of this section, we shall discuss the different mechanisms for distributing the rekeying messages. For each method we will discuss its advantages and disadvantages.

3.5.1 Media-Independent Channel

The first method to convey the rekeying information is to use a channel that is independent of the multimedia content. This can be accomplished in several different ways. First, one could have a security system that is completely separate from the multimedia system, and the key information is transmitted using any other channel that is available to the application. A second manner by which this can be accomplished is through a Systems level operation. In fact, MPEG-2 Systems (ISO/IEC 13818-1) provides the Entitlement Control Messages (ECM) as a means to convey keys associated with scrambling MPEG-2 multimedia streams. The ECM is transmitted as a stream separate from the multimedia. As another

example, the MPEG-4 standard also provides a Systems level data stream to convey security information. In MPEG-4, the Intellectual Property Management and Protection (IPMP) framework provides IPMP Descriptors (IPMP-Ds) and IPMP Elementary Streams (IPMP-ESs) that can help an IPMP system decrypt or authenticate media elementary streams [59]. Both the MPEG-2 ECMs, MPEG-4 IPMP-Ds and MPEG-4 IPMP-ESs can be used to convey rekeying information associated with a multicast service.

Further, many multimedia standards provide data fields that may be used by the system designers to convey non-normative application-specific parameters. For example, in MPEG-1 Video the bistream format for the video sequence layer, the group of pictures layer, and the picture layer provides a mechanism to convey optional *user* data [60]. These fields may be also used to convey security data, such as rekeying messages.

One of the advantages of using the media-independent channel is the ability to assign a delivery protocol to the rekeying messages that is different from the delivery protocols used by the other components of the data stream. Since encryption and decryption keys must be exactly known in order to perform decryption, rekeying messages are extremely sensitive to errors. It is essential that all receivers completely receive a correct rekeying message before the new key takes effect. Without a mechanism to ensure that a rekeying message is received by all legitimate members, some users will be unable to decrypt future content and future rekeying messages.

When the rekeying messages are transmitted using a media-independent channel, their delivery can be performed using a reliable multicasting protocol, such as RMTP and SRM [53,61–63]. However, in addition to using reliable multicasting, it

is necessary to add a feedback mechanism at the application layer. In a multicast security system, it is necessary that the server knows that all users have correctly received the rekeying message before proceeding to the next rekeying message or encrypting the service with the new session key. Therefore, before switching to the new key, the server must wait for an acknowledgement message from each of the clients announcing that they have successfully received the rekeying message.

The use of a media-independent channel can introduce a network security weakness even if there is no cryptographic weakness in the key management scheme. We illustrate this with the following example. When transmitting the rekeying messages in the media-independent mode, the keying messages will be in an encrypted format, such as depicted in (3.24), and kept separated from the other types of data packets. It is possible for an adversary to eavesdrop on the network and observe the presence of these rekeying messages. Even if the rekeying messages are further encrypted by the session key K_s, an eavesdropper on the network may simply observe the rekeying message substream to measure valuable statistical data regarding the multicast membership. For example, if an adversary knows that the key size used is 64 bits and that the rekeying message is of the form (3.24), then when he observes a rekeying message of 64000 bits, he may infer that there are 1000 users in the service. The leakage of statistical information regarding the service membership is a security flaw that can be addressed by using a media-dependent channel. In [64] other system weaknesses were identified that can occur in multicast key distribution schemes even when the underlying cryptographic algorithm is provably correct.

3.5.2 Media-Dependent Channel

A media-dependent channel exists when small amounts of information can be embedded invisibly in the data. In these cases, the rekeying information may be embedded in the content and distributed to those who receive the data [14, 65]. Data embedding, or digital steganography, techniques allow for an information signal to be *hidden* in another signal, known as the cover signal, without dramatically distorting the cover signal. Effective data embedding techniques are those that can invisibly embed data in the cover signal, allow for easy extraction of the embedded information, and achieve a high embedding rate.

Multimedia data types, such as speech, image, and video are well suited for embedding information since introducing a small amount of distortion in their waveforms does not significantly alter perceptual quality [12, 13, 50]. Generic data structures are not well suited for hiding information. The most popular purpose for data embedding is digital watermarking, in which ownership or copyright information is inserted in the cover signal. In this case, the embedding technique must also be robust to attempts to remove or destroy the watermark. Data embedding can also be used to convey side information, such as embedding messages in the content.

Many papers exist on embedding information and watermarks in video. In [50], Hartung and Girod describe a method for inserting digital watermarks into the compressed bitstream of MPEG-2 coded video. They found that they could embed a watermark of 1.25 to 125 bytes/second in NTSC signals. Another method for embedding information in video was presented in [66], and applied to distributing textual information in a video conferencing system. As another example of a scheme with a high embedding rate, a data embedding scheme that is compatible

with standards such as H.263 and MPEG-2 was proposed in [67, 68]. This data embedding technique uses the fractional-pel motion vector as the cover signal for the embedded data, and is able to embed a high bitrate information signal into a video bitstream with an acceptable visual quality degradation. This method for data embedding will be used later in this chapter to demonstrate the feasibility of our multimedia multicast key distribution philosophy.

Associated with many embedding schemes is an *embedding key* which governs how the information is embedded into the cover signal. The size of the embedding key dictates the difficulty for an adversary to attack the embedding rule. For example, in [67, 68], 2 bits of information can be embedded per macroblock, and these 2 bits are embedded by mapping the motion vector to one of 4 regions. There are $4! = 24$ different embedding rules possible. We may therefore associate a 5-bit embedding key K_{emb} with one of these 24 different methods. If a user has the key associated with how the data was embedded, then he may extract the information signal in the multimedia data. An adversary, however, would have to search these 24 possibilities to determine the correct embedding rule to extract the embedded information.

It is desirable to have the size of K_{emb} large in order to make it difficult for an adversary to attack the embedding rule. We now describe the method by which we extend the embedding key size of [67, 68] for use in our later simulations. Suppose that we break the information we wish to embed into 2-bit chunks c_j. We shall choose security parameter q that is a non-negative integer. At random, we shall choose q different embedding rules $(r_0, r_1, \cdots, r_{q-1})$, allowing for repetition in the rules selected. Each embedding rule r_k describes one of the 24 possible ways to map 2 bits to 4 regions. We assign an embedding rule r_k for each chunk c_j according

to $k \equiv j \pmod{q}$. Thus, the $0, q, 2q, \cdots$ 2-bit chunks use embedding rule r_0, the $1, q+1, 2q+1, \cdots$ 2-bit chunks use embedding rule r_1, and so on. The embedding key is thus the concatenation of these rules, which is a key space of 24^q possibilities, and requires $\lceil q \log_2 24 \rceil$ bits to represent. For example, choosing $q = 12$ yields an embedding key size of 56 bits.

The rekeying messages used in either the media-independent or media-dependent cases are almost identical. When using the media-independent approach, only the information needed to update the SK and KEKs needs to be transmitted. However, when using a media-dependent approach, the embedding key must also be updated, requiring that an additional rekeying operation is performed.

The primary advantage of using data embedding to convey rekeying messages compared to a media-independent channel is that data embedding provides an additional layer of security that hides the presence of rekeying messages from potential adversaries. In the conventional approach of using a media-independent channel to convey the rekeying messages, an adversary can observe the external channel and determine information about the membership dynamics of the multicast service, such as the rate at which members join and leave the service as well as being able to infer information about the group membership. From a security point of view, this provides valuable information to a potential adversary. In comparison, data embedding provides *covert* information transferral, whereby the bit rate of the multimedia source is maintained and it is impossible for an eavesdropper to measure information regarding the occurrence of a rekeying operation.

Another effect of the additional layer of security provided by data embedding is the introduction of the embedding key, which must also be maintained by the service provider and stored by the user. A positive benefit of this is that an ad-

versary will not only have to attack the SK and KEKs, but he will also have to attack the key governing the embedding rule in order to acquire rekeying messages. Since the rekeying message is embedded into the multimedia, it is encrypted by the SK, and thereby protected by the SK, the KEK, and the embedding key. For this reason, it is therefore important that the key length of the embedding key is sufficiently long to make it difficult for the adversary to search the embedding key space. We note that a similar increase in protection can be achieved in the media-independent channel by increasing the key length of the session key or by introducing an additional SK. However, encryption algorithms are typically designed for a small set of specified key lengths [69] and it might not be possible to increase the length of the session key.

Finally, when using a media-dependent channel, it is possible to maintain the original data rate of the media without performing the additional computations associated with transcoding [70]. When using a media-independent channel, the rekeying messages introduce additional communication overhead that is in addition to the bandwidth needed to convey the media. In order to keep the data rate of media and rekeying messages identical to the data rate of just the original media, it is necessary to perform computationally intensive transcoding of the media to a lower data rate. However, when using a media-dependent approach to conveying the rekeying messages, the original data rate is maintained, and the data embedding operation provides a graceful degradation of media quality as more data is embedded.

When using media-dependent channels, the issue of reliability becomes more pronounced than in the media-indpendent case since it is not possible to send the rekeying messages through a delivery mechanism separate from the multimedia

data. Since multimedia data is delay sensitive and often transmitted on error-prone channels using *best effort* delivery protocols, it is likely that some media packets will be lost, and the rendering buffer will be filled using the data that successfully arrives. However, when using a media-dependent channel, the lost media packets might contain part of a rekeying message. Since the rekeying messages are embedded in multimedia, which is being delivered through best effort delivery protocols, it is not possible to apply delivery protocols employing retransmissions to improve the reliability of key delivery. There is therefore a tradeoff between covert information transferral and reliability in delivering the rekeying message.

We noted earlier that it is important that the rekeying message is completely received by all users before using the new key. We may, however, address the reliable delivery of the rekeying messages at the application layer. For example, the multimedia system may employ a centralized error recovery technique similar to the NP protocol of [71], however operating at the application layer. The server application takes the k data packets corresponding to a rekeying message and would form h additional parity packets. These $k + h$ packets would be used transmitted as the rekeying message that would be transferred through the media-dependent channel. At the completion of sending the $k + h$ packets, the server would send a message polling the clients whether they were able to successfully decode the rekeying message. The clients would send back acknowledgement messages to the server. If not all of the clients were able to receive the complete rekeying message, the server would employ retransmission, and the process would repeat until all users have successfully received the rekeying message. When all users have received the rekeying message, the server would issue a message instructing all users that it is appropriate to use the new key.

3.6 An Improved Rekeying Message Format

We have described how a rekeying message can be formed during member departure so that each of the remaining members can receive the new root KEK (key encrypting key) by decrypting an appropriate segment of the message using their private KEK. In practice this requires sending additional information that flags to all of the users which segment belongs to which user. Not only does this mean that additional communication overhead is required, but also that sensitive information regarding user identities is released. In particular, adversaries who are members of the service can collect information about other keying messages intended for other users. In order to circumvent this potential weakness, we propose a new format for the rekeying message that is a single, homogenized message from which each user may extract the new root KEK. Such an approach has the advantage that user-specific keying information is not available to other users.

The problem of distributing information simultaneously to multiple users via a single broadcast message while maintaining user anonymity has been previously studied in the literature. Just et al. [9] and Blundo et al. [52] each present a method using polynomial interpolation whereby the broadcast message does not have a partitioned structure like the message in (3.24). A drawback of both of these schemes is that they are suitable for only one transmission, and are not reusable. Specifically, when used to distribute identical information to multiple recipients, each user's secret information is valid for only one transmission, and then is available for other group members to acquire. This is a problem since members may acquire other user's secret information and use this knowledge to enjoy the service after they cancel their membership. In order to use these schemes when the keying material must be updated multiple times, it is necessary to distribute to each user

enough copies of private material to cover the amount of updates needed. Thus, although these schemes use a composite message structure and don't require additional communication overhead for flagging the users, they are not appropriate for applications that require recurrent key distribution.

We therefore desire a scheme that allows for private keying material to be reused while providing a homogenized message form. In Section 3.6.1, we shall describe a new message format that makes use of one-way functions and a broadcast seed to protect each user's private information from compromise [14]. Additionally, although our use of one-way functions can be applied to the polynomial interpolation methods of [9,52], our message format only requires the use of the basic operations of large integer multiplication and modular arithmetic, and does not require the additional functions needed to calculate interpolating polynomials. Then, in Section 3.6.3, we describe how our message format would be used in a tree-based key management scheme to achieve logarithmic usage of communication resources.

First, we introduce parametric (or keyed) one-way functions [58,72], which are the building blocks of our message form.

Definition 2. *A parametric one-way function (POWF) h is a function from $\mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{Z}$ such that given $z = h(x, y)$ and y it is computationally difficult to determine x.*

Parametric one-way functions are families of one-way functions [58,69] that are parameterized by the parameter y. The discrete logarithm provides an example of a POWF since if p is a large prime, and x and y non-identity elements of Z_p^*, the multiplicative subgroup of integers modulo p, it is computationally difficult to determine x given $z = y^x \pmod{p}$ and y [58, 69]. Since symmetric ciphers are typically computationally efficient compared to one-way functions that employ

modular exponentiation, practical one-way functions should be implemented by means of a symmetric encryption cipher. For example, if we let g be a suitable hash function, and E_x a symmetric cipher, then $h(x, y) = g(E_x(y))$ is a POWF. In this case, only ciphers that are secure against known plaintext attacks [58], such as DES or Rijndael, are appropriate. Further, we note that it is not necessary that the hash function g have any cryptographic properties since the required strength is provided by E. Throughout this chapter we shall assume the existence of POWFs that map sequences of $2B$ bits into sequences of B bits.

3.6.1 Basic Message Form

For the basic message form, we shall use the key distribution scheme depicted in Figure 3.1. Suppose that at time $t-2$ the group consists of n users $u_1, u_2, \cdots, u_n$. Each user u_i has a personal B-bit KEK K_i that is known only by the group center and user u_i. Additionally, all of the users share a B-bit root KEK and a session key that will vary with time.

The group center makes available a POWF h that maps a sequence (x, y) of $2B$ bits to B bits. A new function f is defined by prepending a single 1 bit in front of the output of $h(x, y)$, that is $f(x, y) = 1 \| h(x, y)$. The purpose of prepending a bit is to ensure that the modulo operation used by each user will yield $K_\epsilon(t)$.

Suppose, without loss of generality, that user n decides to leave at time $t-2$, then both $K_\epsilon(t-1)$ and $K_s(t-1)$ must be updated. The root KEK is updated first, and then used to encrypt the new session key. In order to update $K_\epsilon(t-1)$, the GC first broadcasts a B-bit random seed $\mu(t)$. Next, the GC forms $K_\epsilon(t)$ and calculates the rekeying message as

$$\alpha_\epsilon(t) = K_\epsilon(t) + \prod_{i=1}^{n-1} f(K_i, \mu(t)). \tag{3.25}$$

A legitimate member u_i may decode $\alpha_\epsilon(t)$ to get the key $K_\epsilon(t)$ by calculating $\alpha_\epsilon(t)$ (mod $f(K_i, \mu(t))$).

We observe that the only property of $\mu(t)$ that is needed is that it is known by all of the recipients. We can therefore achieve a different variation of the scheme by choosing $\mu(t) = K_\epsilon(t-1)$ or $\mu(t) = K_s(t-1)$, which does not require the transmission of the random seed by the system.

We now discuss how this message format reduces the communication overhead compared to a partitioned message format, such as is depicted in (3.24). Current multicast key management schemes, such as [6–8], focus on the size of the payload (the rekeying information), and not on the size of the entire message (including the rekeying message and the header). In fact, the transmission of the messages that flag the users which portion of the message is intended for them can add significant communication overhead when used in conventional tree-based schemes. To illustrate this, we consider the basic key management scheme depicted in Figure 3.1, with $n+1$ users. When using the partitioned message form of (3.24), it is necessary to send a header message that describes the user IDs associated with each of the blocks in the payload rekeying message. Since it requires at least $\log_2(n)$ bits to describe the user IDs for n users, we need an additional overhead of $n\log_2(n)$. Therefore, the percentage of the message size that corresponds to the communication overhead is

$$\rho = \frac{n\log_2 n}{nB_k + n\log_2 n}, \tag{3.26}$$

where B_K is the bit length of the KEK K_ϵ. For large n the communication becomes a significant portion of the message size.

However, the message format of (3.25) is a single, homogenized message that does not require any communication overhead. If we use $\mu(t) = K_\epsilon(t-1)$, then it is

not necessary to broadcast $\mu(t)$ and the total message size of (3.25) is $n(B_K + 1)$, whereas the total message size from the traditional format was $n(B_K + \log_2 n)$. Therefore, as long as $\log_2 n > 1$, the message format of (3.25) is more efficient in terms of communication. This occurs when we are providing service to a group with more than 2 users. Therefore, we have made a tradeoff between communication and computation. The message format of (3.25) uses less overall communication at an expense of requiring more computation to form the message.

3.6.2 Security Analysis of Residue-based Method

The residue-based method for multicast key distribution was described in Section 3.6. The basic form of rekeying message in the residue-based method is

$$\alpha = X + \prod_{j=1}^{r} Y_j, \tag{3.27}$$

where X and Y_j are drawn uniformly from the set $\{1, 2, \cdots, B\}$. The variable X corresponds to the secret, or the key, that is being convey, while the Y_j are the user-specific shares that mask the secret.

This section describes an information theoretic investigation of the security that this method provides for protecting X, and some motivation for using one-way functions with a time-varying seed.

Information Theoretic Analysis

Consider the scenario where there is only one Y term, and define the random variable $Z = X + Y$. In general, the entropy of Z is difficult to relate to the entropy of two arbitrary random variables X and Y. The following relationship holds

$$H(Z) \leq H(X, Y) \leq H(X) + H(Y), \tag{3.28}$$

and in general the bound need not be met. In particular, the consider the random variables

$$X = -Y = \begin{cases} 1 & : \text{ with probability p}(1) = 1/2 \\ 0 & : \text{ with probability p}(0) = 1/2 \end{cases} \tag{3.29}$$

In this example, $H(X) = H(Y) = 1$, while $Z = 0$ so that $H(Z) = 0$. The following lemma places a lower bound on the entropy of Z.

Lemma 8. *Suppose that $Z = X + Y$, then $H(Z|X) = H(Y|X)$.*

Proof.

$$\begin{aligned} H(Z|X) &= \sum_x p(x) H(Z|X = x) \\ &= -\sum_x p(x) \sum_z p(Z = z|X = x) \log p(Z = z|X = x) \\ &= -\sum_x p(x) \sum_y p(Y = z - x|X = x) \log p(Y = z - x|X = x) \\ &= \sum_x p(x) H(Y|X = x) \\ &= H(Y|X). \end{aligned}$$

□

Thus, $H(Z) \geq H(Z|X) = H(Y|X) = H(Y)$, since X and Y are assumed independent. Similarly, $H(Z) \geq H(X)$.

The pdf of $Z = X + Y$ is simply the convolution of the pdf of X and the pdf of Y. Now, when X and Y are uniformly drawn from $\{1, \cdots, B\}$, then the pdf of Z is a triangular function, and the entropy of Z may be calculated directly. Suppose that $h(k) = B^2 p_Z(k)$, then the entropy of Z is

$$H(Z) = -\frac{1}{B^2}\left[\left(\sum_{k=2}^{2B} h(k) \log h(k)\right) - \left(2 \log B \sum h(k)\right)\right] \tag{3.30}$$

$$= -\frac{1}{B^2}\left[\sum h(k) \log h(k)\right] + \frac{2 \log B}{B^2} \sum h(k) \tag{3.31}$$

$$= -\frac{1}{B^2}\left[\sum h(k) \log h(k)\right] + 2 \log B. \tag{3.32}$$

Due to the symmetry of the triangle function $h(k)$,

$$\sum_{k=2}^{2B} h(k) \log h(k) = 2 \left(\sum_{j=1}^{B-1} j \log j \right) + B \log B. \tag{3.33}$$

Thus

$$\begin{aligned} H(Z) &= -\frac{2}{B^2} \left(\sum_{j=1}^{B-1} j \log j \right) - \frac{1}{B} \log B + 2 \log B & (3.34) \\ &= \left(2 - \frac{1}{B} \right) \log B - \frac{2}{B^2} \left(\sum_{j=1}^{B-1} j \log j \right). & (3.35) \end{aligned}$$

The entropy of the sum was calculated for X and Y drawn uniformly from a range $\{1, 2, \cdots, B\}$, where $B = 2^b$ and is recorded in Table 3.1. Examining this table reveals that the difference between the entropy in Z and the entropy of either X or Y tends toward an asymptotic limit. The exact value and significance of this limit is currently not known.

The security of the residue-based method is measured by the uncertainty that remains in the key given only the observation of the rekeying message. For the case of a single term in the product, this is measured by the entropy $H(X|Z)$. The following lemma relates $H(X|Z)$ to the entropies $H(X)$, $H(Y)$, and $H(Z)$.

Lemma 9. *Suppose X and Y are independent, and $Z = X + Y$, then*

$$H(Z|X) = H(X) + H(Y) - H(Z). \tag{3.36}$$

Proof. By application of the chain rule, $H(X|Z) = H(X, Z) - H(Z)$. Observe that there is a unique correspondence between the joint variable (X, Z) and (X, Y). Therefore, $H(X, Z) = H(X, Y) = H(X) + H(Y)$. Substitution gives the desired result. □

Applying this lemma with the values presented in Table 3.1 gives that $H(X|Z) \approx H(X) - 0.721347$. This result implies that roughly one bit of security is lost when

b	H(Z)	b	H(Z)
1	1.5000000000000	13	13.7213474774632
2	2.6556390622296	14	14.7213475090783
3	3.7023191426459	15	15.7213475174478
4	4.7159395672686	16	16.7213475196565
5	5.7198327831914	17	17.7213475202379
6	6.7209281467435	18	18.7213475203902
7	7.7212325045380	19	19.7213475204300
8	8.7213162233391	20	20.7213475204415
9	9.7213390603855	21	21.7213475204434
10	10.7213452464840	22	22.7213475204440
11	11.7213469122180	23	23.7213475204448
12	12.7213473584537	24	24.7213475204435

Table 3.1: The entropy of the sum $Z = X+Y$, where X and Y are drawn uniformly from integers between 1 and $B = 2^b$.

there is only a single Y term. Thus, an adversary must only search a keyspace that is half as large as the original keyspace.

If the original keyspace is sufficiently large, then this reduction might not be significant. For example, searching a keyspace of 100 bits is effectively as difficult as searching a keyspace of 99 bits.

The security of this scheme when more Y_j terms are used remains to be investigated. It is conjectured that the amount of bits lost will increase since the distribution of $Y = Y_1 Y_2 \cdots Y_r$ will no longer be uniform. Additionally, the exact value of the 0.721347 term remains to be explored. Finally, since direct calculation

of the entropies for large $B \approx 2^{20}$ takes considerable computing effort, it would be desirable to construct bounds on the entropy of $Y = Y_1 Y_2 \cdots Y_r$, as well as on the entropy of $Z = X + Y$.

Attacks by Insiders

We now examine the possibility for a member of the group to attack the security of the system by gathering or inferring information not intended for them. Suppose that the basic form of the rekeying message is

$$\alpha(t) = X(t) + Y = X(t) + \prod_{j=1}^{r} Y_j, \tag{3.37}$$

where $X(t)$ denotes the time-varying key that the GC is distributing to the users. The Y_j are user specific secrets that allow a user to determine $X(t)$ given α by performing a modulo operation. In a dynamic environment, it is important to prevent members from acquiring other member's secrets. In the basic form of the rekeying message, once the user u_s has determined X, he may determine $\prod_{j \neq s} Y_j$ by

$$\prod_{j \neq s} Y_j = \frac{\alpha(t) - X(t)}{Y_s}. \tag{3.38}$$

This allows user u_s to depart the system, receive a future rekeying $\alpha(t)$, and use $\prod_{j \neq s} Y_j$ in the modulo operation to determine future $X(t)$.

It is therefore necessary to make the Y term also time-varying in order to make it more difficult to acquire $X(t)$. An initial approach to solve this problem was to define $Y(t) = \lambda(t) \prod_{j=1}^{r} Y_j$. In this case, an inside adversary is able to calculate

$$A(t) = \lambda(t) \prod_{j=1}^{r} Y_j \tag{3.39}$$

Since the $\lambda(t)$ are chosen at random, one might expect that this would introduce enough randomness to make calculating $\prod_{j=1}^{r} Y_j$ difficult. This, however, is not the

case.

Consider the probability that two random integers are relatively prime. A non-rigorous derivation of this probability would proceed as follows. The probability that one number is divisible by a prime p_i is $1/p_i$. If both numbers were chosen independently of each other, the probability that both are divisible by p_i is $1/p_i^2$ and hence the probability that they are not both divisible by p_i is $(1-1/p_i^2)$. The probability that two numbers are coprime can then be estimated by

$$W_2 = \prod_{p_i}\Big(1 - \frac{1}{p_i^2}\Big). \tag{3.40}$$

In order to calculate W_2 it is easier to start with $1/W_2$.

$$\frac{1}{W_2} = \prod_{p_i}\Big(\frac{1}{1-\frac{1}{p_i^2}}\Big). \tag{3.41}$$

By expanding $\frac{1}{1-x}$ into a series expansion and observing that the product results in a term for every integer, we get

$$\frac{1}{W_2} = \sum_{n=1}^{\infty}\frac{1}{n^2} = \zeta(2) = \frac{\pi^2}{6} \tag{3.42}$$

where ζ is Riemann's zeta function [73]. Therefore, the probability of two numbers being relatively prime is $W_2 = \frac{6}{\pi^2}$. This can be extended to the probability that s numbers are relatively prime. Let W_s be the probability that s non-negative integers are relatively prime, then by the same idea as before

$$W_s = \prod_{p_i}\Big(1 - \frac{1}{p_i^s}\Big) \tag{3.43}$$

which leads to

$$\frac{1}{W_s} = \sum_{n=1}^{\infty}\frac{1}{n^s} = \zeta(s). \tag{3.44}$$

We tabulate the first few such probabilities in Table 3.2.

W_2	0.608
W_3	0.832
W_4	0.924
W_5	0.964
W_6	0.983

Table 3.2: Probabilities of Coprimality

This result states that an inside adversary, after gathering m observations $A(t_m)$ has an increasingly likely chance of calculating $\prod_{j=1}^{r} Y_j$. In fact, with just 8 observations, there is over a 99.5% chance that he will be able to calculate $\prod_{j=1}^{r} Y_j$.

In a dynamic scenario, calculating $\prod_{j \neq s} Y_j$ does not guarantee being able to acquire future $X(t)$. However, the Y_j of any other member who remains in the service will do. Unlike other cryptographic methods, such as RSA, where the factors have greater than 500 bits, each Y_j is typically less than a couple hundred bits and factoring the product $\prod_{j=1}^{r} Y_j$ will not be too difficult [58,69]. In this case, the adversary's task is to reconstruct a Y_j given a list of factors of $\prod_{j=1}^{r} Y_j$. Since there is no guarantee how many factors each Y_j will have, it is not reasonable to rely on the difficulty of recombining factors to protect the key $X(t)$.

Other approaches to making the Y term time-varying have been examined. In order for the time-varying form $Y(t)$ to be secure, it must be difficult to calculate an individual Y_j term given knowledge of the rekeying message $\alpha(t)$ and the key $X(t)$. A natural approach for making it difficult to calculate a value y given a value $g(y)$ is to make g a one-way function. The idea of using one-way functions provides the difficulty needed to prevent an inside member from calculating another user's private key Y_j. However, using one-way functions alone did not provide security

for protecting $X(t)$. As mentioned earlier, it is necessary to make the Y term time-varying, and based on the arguments presented above, one needs to make the product term $\prod_{j=1}^{r} Y_j$ time-varying. This was accomplished by introducing the time-varying broadcast seed.

By broadcasting $\mu(t)$ and using a non-reversible function f, the adversary is instead able to calculate

$$A_i = \prod_{j \neq i} f(K_j, \mu(t)). \tag{3.45}$$

Factoring A_i provides information about $f(K_j, \mu(t))$. Since it is difficult to acquire K_j given $\mu(t)$ and $f(K_j, \mu(t))$, the private user information is protected. At the next time instant, when $\mu(t+1)$ is broadcast, the adversary's knowledge of $f(K_j, \mu(t))$ does not help him in calculating $f(K_j, \mu(t+1))$, and he can extract K_ϵ only if he has the needed keys assigned to him.

3.6.3 Achieving Scalability

When the multicasting group is very large, it is necessary to make efficient usage of communication resources. Improved resource scalability can be achieved by employing a tree-based key management scheme to update the SK and KEKs [6,7].

A binary tree is shown in Figure 3.5, though in the general case the tree can be an a-degree tree. Attached to the tree above the root node is the session key K_s. Each node in the tree is assigned a KEK called an internal key (IK) which is indexed by the path leading to itself. The symbol ϵ is used to denote empty string, which is the path of the root node to itself. Each user is assigned to a leaf and is given the IKs of the nodes from the leaf to the root node in addition to the session key. For example, user u_{111} is assigned keys K_{111}, K_{11}, K_1, K_ϵ, and K_s. All of the keys, with the possible exception of the leaf keys, may vary with time to reflect

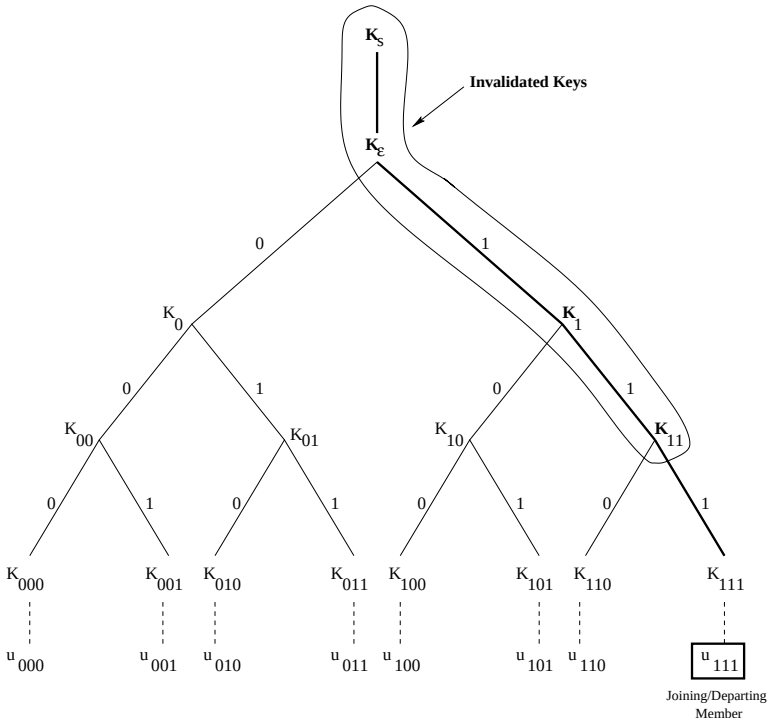


Figure 3.5: Tree-based key distribution.

the changing dynamics of the group membership.

During periodic refreshes, only the session key needs to be updated, and the same protocol as presented in Section 3.4.1 can be used. We will now address how to operate during additions and deletions of members.

The GC is in charge of keeping track of the group members, and assigning them to positions on the tree. Although it is easiest to have the membership tree be a balanced tree, it is not necessary. For example, in [54], a non-balanced tree employing one-way functions is used in a key management scheme allowing member joins and departures is used. In this work, we shall just describe the procedure for adding members to a non-full balanced tree, and removing members from a full balanced tree. If a balanced tree is full, meaning all of the leaf nodes have members associated with them, then it is necessary to spawn a new layer of nodes

when adding members. Additionally, by following the example of Balenson et al. [54] one can see how to make an approach handle member joins and departures for non-balanced trees.

Member Join

The member join operation does not involve the message format of (3.25) since each node of the key tree updates itself. Nonetheless, we present this case for completeness. Consider the binary tree depicted in Figure 3.5, that has 7 members u_{000} through u_{110}. If user u_{111} would like to join the group, the keys on the path from his leaf node to the tree's root as well as the SK, must be changed in order to prevent access to previous communications. Thus new $K_\epsilon(t)$, $K_1(t)$, $K_{11}(t)$ and $K_s(t)$ must be generated by the GC. The key encrypting keys can be updated from top to bottom by using $K_\epsilon(t-1)$ to encrypt $K_\epsilon(t)$, $K_1(t-1)$ to encrypt $K_1(t)$, and $K_{11}(t-1)$ to encrypt $K_{11}(t)$. Thus, all users can acquire the new root KEK, while only members u_{100}, u_{101}, and u_{110} can acquire $K_1(t)$. After updating the KEKs, the session key is updated by encrypting with the new root KEK $K_\epsilon(t)$.

Member Departure

When a member leaves the group, multiple keys become invalidated because that user shares these keys with other users. For example, in Figure 3.5, user u_{111} shares K_{11} with user u_{110}. Thus, if user u_{111} departs the multicast group, the key encrypting keys K_{11}, K_1, and K_ϵ become invalidated. These keys must be updated. Observe that K_{111} does not need to be updated since it is a private key and is not shared with any other users.

There are two basic approaches to updating the keys during a member depar-

ture: update the keys from the root node to leaf nodes, or from leaf nodes to root node. In the first approach, the *top-down* approach, when user u_{111} departs, the keys are updated in the order K_ϵ, K_1, and K_{11}. The second approach, the *bottom-up* approach, updates the keys in the order K_{11}, K_1, and K_ϵ. After updating the key encrypting keys, the root KEK $K_\epsilon(t)$ can be used to encrypt the new session key $K_s(t)$ and a single message may be broadcast to all members.

Let us focus on how to update these keys using the top-down approach in conjunction with the new message form when user u_{111} departs. First, a random seed $\mu(t)$ is broadcast to all members, or some shared information, such as $K_\epsilon(t-1)$ is used as $\mu(t)$. Next, the root KEK $K_\epsilon(t-1)$ will be updated. In order to do this, the message

$$\alpha_\epsilon(t) = K_\epsilon(t) + f(K_0(t-1), \mu(t)) f(K_{10}(t-1), \mu(t)) f(K_{110}, \mu(t)) \tag{3.46}$$

is formed and broadcast. Next, $K_1(t-1)$ is updated by forming the message

$$\alpha_1(t) = K_1(t) + f(K_{10}(t-1), \mu(t)) f(K_{110}(t-1), \mu(t)) \tag{3.47}$$

and broadcasting. The last KEK to update is $K_{11}(t-1)$. This can be done by sending the message

$$\alpha_{11}(t) = K_{11}(t) + f(K_{110}(t-1), \mu(t)). \tag{3.48}$$

Upon updating the KEKs, the session key may then be updated. To do this, the root KEK is used to encrypt $K_s(t)$ and the resulting message is broadcast.

In order to update the keys from a bottom-up approach, the random seed is broadcast, and then $K_{11}(t-1)$ is updated via

$$\alpha_{11}(t) = K_{11}(t) + \prod_{j=0}^{0} f(K_{11j}(t-1), \mu(t)) \tag{3.49}$$

The next key that is updated is $K_1(t-1)$. Since the two users beneath K_1 share a common key that is not invalidated by the departure of member u_{111}, we may reduce communication and computation by using this key to update K_1. The resulting message

$$\alpha_1(t) = K_1(t) + \prod_{j=0}^{1} f(K_{1j}(t), \mu(t)) \tag{3.50}$$

is broadcast. Since $K_{10}(t-1)$ is still valid, we implicitly updated $K_{10}(t) = K_{10}(t-1)$. To update $K_\epsilon(t-1)$ we may use the new key $K_1(t)$ as well as the old key $K_0(t) = K_0(t-1)$ and form the message

$$\alpha_\epsilon(t) = K_\epsilon(t) + \prod_{j=0}^{1} f(K_j(t), \mu(t)). \tag{3.51}$$

Finally, the session key is updated by encrypting the new session key $K_s(t)$ using the new root KEK $K_\epsilon(t)$, and broadcasting the message

$$\alpha_s(t) = E_{K_\epsilon(t)}\left(K_s(t)\right). \tag{3.52}$$

The amount of multiplications as well as the communication requirements needed to update all of the KEKs using the top-down approach and the bottom-up approach will differ. Assume that we have n users and keys assigned to each of these users using an a-ary tree. If the tree is a full, balanced tree with $L = \log_a n$ levels, then the amount of multiplications needed to update the KEKs during a member departure using a top-down approach is:

$$\begin{aligned} C_{td} &= \sum_{i=1}^{L} i(a-1) \\ &= (a-1)\frac{\log_a n(log_a n + 1)}{2}. \end{aligned} \tag{3.53}$$

Similarly, the amount of multiplications needed to update the KEKs using a bottom-up approach is

$$C_{bu} = aL - 1$$

$$= \quad a \log_a n - 1. \tag{3.54}$$

The amount of communication needed for each of these schemes is directly related to the amount of multiplications performed. If each internal key is B bits long, and a rekeying message requires M multiplications, then the message size will be $M(B+1)$ bits. Therefore, the bottom-up approach to renewing the keys requires less computation and communication. However, if the SK needs to be updated sooner, one may wish to use a top-down approach since it allows one to update the root KEK first, the session key next, and finally the remaining IKs.

3.7 System Feasibility Study

In this section, we study the issues related to the feasibility of using a key management system for multicast multimedia. When designing a cost effective system, one must consider the balance between computation, communication, and storage resources.

One of the primary advantages for using a tree-based key distribution scheme is that it achieves good scalability in the amount of communication needed to update the network. The need for using a tree-based key distribution scheme becomes more pronounced as the group size increases. If the group size is small, for example less than 10 users, there might not be any benefit from using a tree-based key distribution scheme, and one might want to consider the simple key distribution scheme presented in Section 3.6. However, the $\mathcal{O}(\log n)$ communication needed by most tree-based schemes makes the use of a tree-based scheme essential when the group size is several thousand or more users.

Another issue that should be considered is the amount of storage needed by

the GC and each individual user. If each user has extremely limited storage, then the simple distribution scheme of Section 3.6 might be appropriate. However, although a tree-based scheme may require more storage for each user, and a factor more storage for the GC, typically this is not as important of a consideration as communication resources.

As an example, in the scheme presented in Section 3.6.3, the amount of multiplications (computation) needed to update the KEKs for the bottom-up approach was calculated to be $C_{bu} = a \log_a n - 1$. The communication needed is proportional to the amount of computation needed. The amount of storage needed by the GC to keep track of the KEKs is

$$S = \frac{a^{L+1} - 1}{a - 1} \tag{3.55}$$

keys, while the amount of storage needed by each user is $\log_a n + 2$ keys.

Next, one must consider the channel that one is transmitting the keys across. Whether transmitting via an external channel or an internal channel, there is a channel rate that governs how quickly the keying information may be distributed. For example, suppose we are transmitting the rekeying information for the scheme of Section 3.6.3 via an internal channel. If we denote R as the embeddable channel rate (in bits/second), B_{KEK} to be the key length of a KEK, B_s to be the key length of the session key, B_μ the bit length of the random seed $\mu(t)$, and B_{emb} to be the key length governing the data embedding rule, then the amount of time needed to update the entire system of keys is

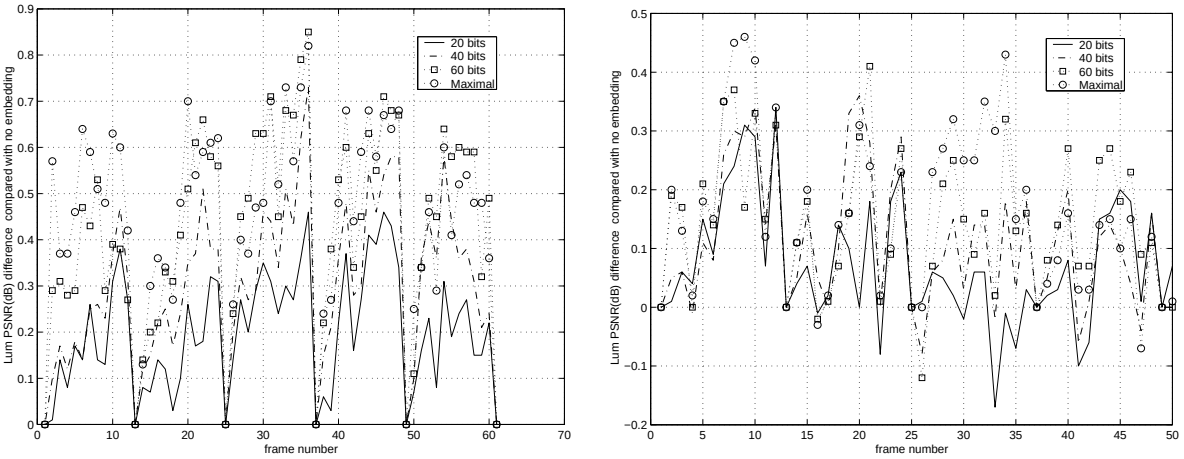
$$T = \frac{C_{bu} B_{KEK} + B_s + B_{emb} + B_\mu}{R}. \tag{3.56}$$

Since T is related to the bit size of each of the keys, it is therefore related to the security levels protecting the service. This amount of time corresponds to the amount of time the departing member may still enjoy the service before no

longer being able to decode the video stream. If we desire to increase the level of protection of the multimedia, then B_s must be increased, which leads to an increase in the amount of time needed to refresh the entire set of keys. Similarly, if we desire to increase the difficulty an adversary would have in decoding rekeying messages, then we need to increase B_{KEK}, which would also increase T.

In designing a system, these tradeoffs must be weighed and considered from a realistic point of view. Although it might be desirable to have extreme protection of the content, in a dynamic group, it is not realistic that it take an hour to update the set of keys.

To demonstrate these considerations, we present some simulation results using the data embedding scheme proposed in [68]. The degradation of the visual quality when different amounts of bits embedded per frame were measured for the *Foreman* and *Miss America* QCIF video sequences. The H.263 TMN-11 video codec was used with annexes D, I, J, F turned on [74]. The bitrate in the simulation is 64kbps with a frame rate 10f/s, and every 12th frame is INTRA coded. The peak signal-to-noise ratio (PSNR) of luminance component with different data embedding rates are compared with the PSNR of luminance without embedding. In the simulations, the four cases compared correspond to when the number of bits embedded in a P-frame is upper bounded by 20, 40, 60 and no constraint (maximal). The PSNR differences are shown in Figure 3.6(a) for *Foreman* and Figure 3.6(b) for *Miss America*. Their average PSNR differences are also listed in Table 3.3. In all cases, the PSNR degradation of Luminance is within 1dB for both *Foreman* and *Miss America*, which normally cannot be detected by human visual system for video applications. Additionally, it was shown in [68] that data embedding at half-pel motion estimation at most degenerates the video coding



(a) (b)

Figure 3.6: The peak signal-to-noise ratio (PSNR) difference of the luminance components between no embedding and the embedding scheme of Song et al. with variable embedding rate. (a) Foreman, (b) Miss America.

performance back to integer-pel motion estimation without data embedding.

Using this data embedding scheme in conjunction with the bottom-up approach to member departure discussed earlier, we calculated the amount of time needed to refresh the entire network of keys for a tree of degree $a = 2$, and $n = 2^{20}$ or roughly one million users. We took $B_{KEK} = 56$ bits, $B_s = 56$ bits, $B_\mu = 56$ and $B_{emb} = 20$ bits as the bit lengths for the various keys. These values for B_{KEK}, B_s and B_μ were chosen since they correspond to the key size of the popular

	20 bits	40 bits	60 bits	Maximal
Foreman	0.2002(dB)	0.3054(dB)	0.4264(dB)	0.4477(dB)
Miss America	0.0720(dB)	0.1098(dB)	0.1434(dB)	0.1602(dB)

Table 3.3: Average PSNR difference.

block cipher DES. The resulting times needed to refresh the keys are presented in Figure 3.7. The curves illustrate the inverse relationship with the amount of bits embedded per frame. Using these curves, one can determine the necessary embedding rate needed to refresh the keys in time T. For example, if we have a video service of QCIF images with a frame rate of 20 frames/second, and desire to refresh the keys during member departure in $T = 5$ seconds, then 25 bits must be embedded per frame. In particular, for an embeddability rate of 25 bits/frame, we note that average PSNR difference of the two test sequences is less than 1dB and therefore would introduce no noticeable distortion to the video quality. Further, in video applications that use higher-resolution video formats, such as CIF and SIF format, less distortion occurs for the same embeddability rate. Thus, for the same amount of distortion in video with a larger image size, it becomes possible to rekey larger group sizes, refresh keys faster, or increase the protection by using larger key lengths.

3.8 Extensions to Multilayered Services

In many application environments, the multimedia data is distributed in a multi-layered form. For example, in an HDTV broadcast, users with a normal TV receiver can still receive the current format, while other users with a HDTV receiver can receive both the normal format and the extra information needed to achieve HDTV resolution. As another example, the MPEG-4 standard allows for multiple media streams corresponding to different object planes to be composited. In either of these cases, it will be desirable for service providers to separately control access to the different layers of media. The key management schemes must therefore be considered separately, yet incorporate new key management functionalities that are

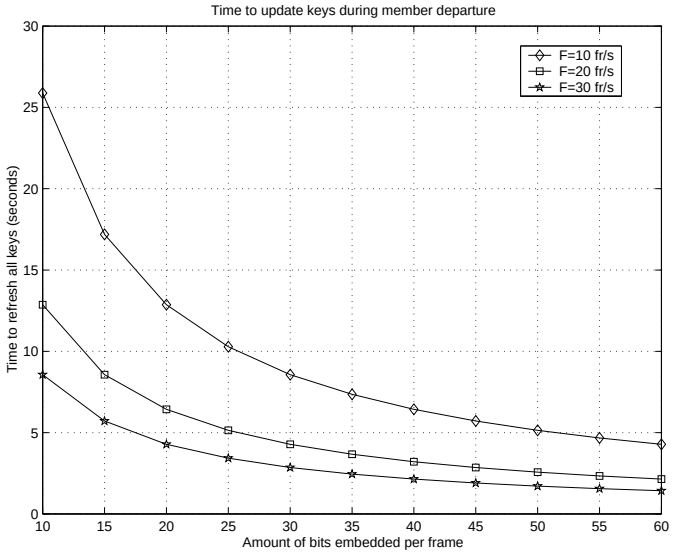


Figure 3.7: The time needed to refresh the entire set of keys during a member departure using the bottom-up approach with different frame rates F, and different amounts of bits embedded per frame. The group size is $n = 2^{20}$, or roughly one-million users.

not present in conventional multicast key management schemes. 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 other layers. Hence multi-layered, or multi-object multimedia services will require additional functionality added to a multicast key management scheme.

As an example of the additional functionality needed, we use our tree-based scheme of Section 3.6.3 and consider the problem of managing keys for two levels of service corresponding to a low quality and high quality service. Extensions to more layers or objects is straight forward.

Suppose the multimedia data stream consists of two layers, which are denoted as D^l and D^h. D^l provides the low resolution service only, while high-quality service

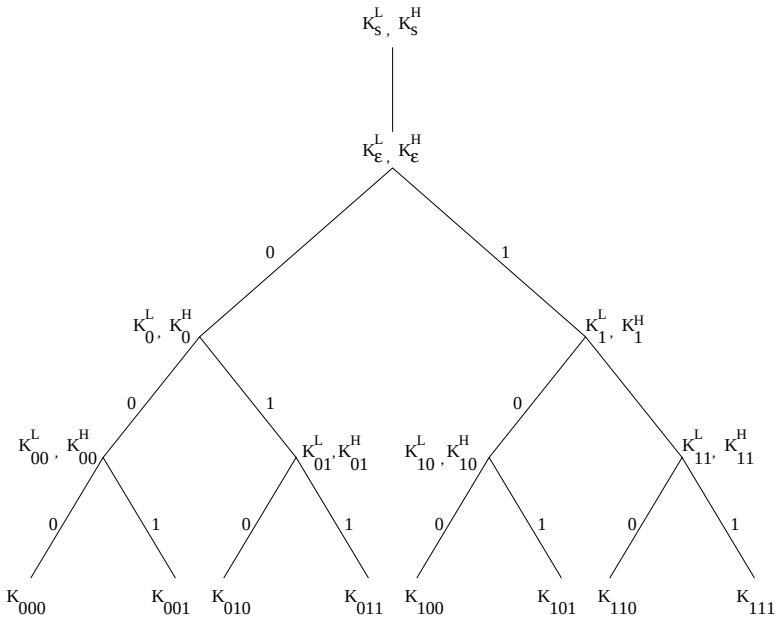


Figure 3.8: Key distribution for multi-layer multimedia multicast.

can be obtained by receiving both the base-layer D^l and the refinement-layer D^h. The GC will have two session keys $K^l_s(t)$ and $K^h_s(t)$. $K^l_s(t)$ is used to encrypt D^l and $K^h_s(t)$ is used to encrypt D^h. Similarly, each internal node in the key tree has two internal keys $K^l_\sigma(t)$ and $K^h_\sigma(t)$, where σ is the index of the nodes in the tree. Group members who want to receive the lower quality service will be assigned the low-layer session key, as well low-layer keys from the root to the leaf which stands for this member. Group members who want to receive high quality service will be assigned both the low-layer and high-layer keys. The rekeying scheme is similar to the one layer case described earlier, but requires additional functionalities since users may switch between the different levels of service.

- **Refreshing the low-quality session key:** The new session key associated with the low-quality level may be refreshed by encrypting with the root low-quality KEK $K^l_\epsilon(t)$ and transmitting the message $\alpha^l_s(t) = E_{K^l_\epsilon(t)}\left(K^l_s(t)\right)$.

- **Refreshing the high-quality session key:** The procedure for refreshing of the high-quality session key is identical to the procedure for refreshing the low-quality session key, but using $K_s^h(t)$ and $K_\epsilon^h(t)$ instead.

- **New member joins low-quality service:** A new member may desire to join the low level service. In this case, the low-quality session key and IKs must be renewed, which can be done by applying the procedure of Section 3.6.3.

- **New member joins high-quality service:** A new member may desire to join the high level service. In this case, both the low-quality and high-quality keys must be renewed. To do this, the procedure of Section 3.6.3 is applied twice, once for the low-quality keys, and once for the high-quality keys.

- **High-quality user leaves the group:** In this case, both session key $K_s^l(t-1)$ and $K_s^h(t-1)$ and corresponding IKs for both D^l and D^h have to be changed. This can be done using the algorithms in Section 3.6.3 twice.

- **Low-quality user leaves the group:** In this case, only session key $K_s^l(t-1)$ and corresponding IKs for base-layer D^l needs to be changed, which can be done using the algorithms in Section 3.6.3 once on the appropriate low-layer keys.

- **Low-quality user changes to high-quality:** In this case, the high-layer SK $K_s^h(t-1)$ as well as the high-layer IKs must be changed has to be changed to prevent the user from accessing the past high quality service. The new SK $K_s^h(t)$ and IKs keys from root to the leaf are directly given by the GC to this user during registration to the new level of service.

- **High-quality user change to low-quality:** The session key $K_s^h(t-1)$ and corresponding IKs for high-layer have to be changed to prevent this user from accessing the future high quality information. This can be done using the algorithms in Section 3.6.3 once on the high-layer internal keys.

3.9 Chapter Summary

The secure distribution of multimedia multicasts necessitates the distribution and management of keying material. In this chapter, we have examined the problem of managing keys needed to secure multimedia multicasts. We presented a new format for the rekeying messages associated with multicast key management, as well as described two modes of conveyance for transmitting the rekeying messages.

We began by discussing the fundamental problem of securely distributing information simultaneously to a group of users. This fundamental problem is at the heart of multicast key management schemes, where the information to be distributed is a new session key. We examined a simple key management scheme to motivate the importance of reducing the communication overhead associated with identifying which portion of a rekeying message is associated with each user. The communication overhead is reduced by using a homogenized message format from which every user can perform a suitable operation to extract the new keying information. We presented a homogenized message format, built using one-way functions and large integer arithmetic, that allows for each user to perform a modular operation to extract the new key information. We then examined the security of the residue-based rekeying message from an information theoretic perspective, and also showed that the residue-based rekeying message format is resistant to attacks by members of the service attempting to acquire private keying information

of other members.

Typically, the information associated with rekeying is distributed via a media-independent channel. However, multimedia data allows for a media-dependent channel, such as is provided by data embedding techniques. By embedding the keying information in the multimedia content, the key updating messages associated with secure multicast key management schemes may be hidden in the data and used in conjunction with encryption to protect the data from unauthorized access. The primary advantage of using data embedding to convey rekeying messages compared to the traditional use of a media-independent channel is that data embedding hides the presence of rekeying messages from potential adversaries, thereby making it more difficult for eavesdroppers to measure information regarding membership dynamics. Further, the use of data embedding allows the application to maintain the data rate of the media without performing computationally expensive transcoding operations.

We used our proposed message form in conjunction with a data embedding technique for block-based motion compensated video compression to illustrate that the amount of time needed to update the entire network of keys is related to the amount of users in the service, key lengths used, and the embeddable channel rate. For a video service providing QCIF images with a frame rate of 20 frames/second, we observed that it was possible to refresh the keys for a group size of roughly one million users in 5 seconds when we used an embeddability rate of 25 bits/frame. The distortion introduced to the video sequence was less than 0.8dB of PSNR and was not perceptible. Finally, by adding extra functionality to multiple key trees, multicast key distribution schemes can be extended to protect multiple layers of multimedia content in an efficient manner. The additional operations needed

to manage the keys for multilayered services is more complex than traditional multicast services since users may switch between different levels of service. We presented an example of a key management scheme for two levels of service, and described the necessary operations needed to allow users to drop from a high-quality service to a low-quality service, and also upgrade their service from a low-quality to a high-quality service.

In the next chapter we build a protocol suitable for rekeying users in a dynamic multicast service. The protocol will employ both a homogenized message format and a tree-based key hierarchy in order to achieve desirable scalability of communication resources.

Chapter 4

A Key Management Architecture for Conditional Access Systems in Dynamic Multicasting Scenarios

4.1 Introduction

With the advancement of networking technologies, such as broadband IP and satellite networks, many opportunities have been created for the delivery of bandwidth intensive media such as audio and video. Many of these future multimedia applications will involve group-based scenarios, where users may join and leave at anytime. Multicast communication is the most suitable method for delivering data to groups of users due to its efficient usage of network resources. Over the Internet, for example, the recipients of a group communication are associated with a Class D IP address, and may receive messages sent to that address [75]. A server that desires to send communication to the group addresses messages with the group address and transmits a single copy of the message. It is the responsibility of the network and the multicast-enabled routers to deliver the message to the users. By sending only a single copy of the message on the network, the usage of server-side resources

such as bandwidth and processing is reduced.

The adaptation of multicast into commercial applications depends on the ability to control access to the communications. For example, consider a service provider that distributes streaming content, such as multimedia streams, to a group of paying users via a multicast technology. In such an application, the service provider must be able to ensure the availability of multicast data to privileged members while preventing unauthorized use of this data by non-privileged users. A service provider may control access to content by encrypting the content using a key that is shared by all group members. The problem of access control is made more difficult when the content is being distributed to a group of users since the membership will most likely be dynamic, with users joining and leaving the service for a variety of possible reasons. Upon changes in the membership, it is necessary to change the keys associated with the service.

A conditional access system for multicasting must be able to cope with the demands of the application. These demands must not only address the security and access requirements of the service provider, but also address the convenience and satisfaction of the client. Below we have listed several functionalities that are desirable in a conditional access system for dynamic multicast scenarios:

1. *The solution should be able to refresh the keys used to protect content.*

   Due to the bulk quantities of data being multicast, it is feasible that session keys may become compromised. Therefore, it is important that there is a means available to refresh the session key and intermediate keying material in order to maintain a desirable level of content protection.

2. *The solution should provide the ability for members to join and depart the service at will, as well as allow the content distributor to easily revoke a*

member's ability to access content.

Unlike unicast communication, the departure of a group member does not imply the termination of the communication link. In addition, upon departing the service, users must be de-registered and prevented from obtaining future multicasts. Similarly, when new members join the service, it is desirable to prevent them from accessing past content. Additionally, situations might arise where the content provider desires to prevent a user from accessing future content.

3. *The solution should be resistant to member collusion.*

   No subset of the members should be able to collude and acquire keying information of non-colluding members.

4. *The solution should provide a means for an end-user to recover from missed rekeying messages.*

   In many application environments, the connection between a client and the server may be severed. For example, in cellular applications, a client might move temporarily through a region of severe fading. Adverse communication conditions and common accidents, such as a system crash, might mean that the client misses several rekeying messages needed to update his key database. Users might also desire to switch from terminal to terminal, with the possibility of not being able to receive communication while moving across terminals. It is important to have a means that allows the client to resume access to the service.

5. *The solution should allow the user to temporarily transfer access rights to another party.*

In many business scenarios, a client will subscribe to a service where content, such as multimedia or stock quotes, is streamed. Users may wish to transfer their access rights to the data stream to their friends without canceling or transferring their subscription.

6. *The solution should address the issue of resource scalability for scenarios consisting of large privileged groups.*

   In many applications, the size of the group may be very large and possibly on the order of several million users. The required communication, storage, and computational resources should not become a hindrance to providing the service as the group size increases.

In this chapter we present an architecture for the management of keys in a conditional access multicast system. The system that we describe makes use of a tree-structured key hierarchy and basic primitive operations to provide a solution that satisfies the above requirements. Additionally, whereas most of the multicast key management schemes in the literature do not consider the issue of flagging to the user which rekeying messages are intended for them, we provide this important functionality in our message structure. We next focus on the usage of communication resources and calculate the amount of communication needed to perform a member join and a member departure operation for different tree degrees and different amount of users. We determine the optimal tree degree for scenarios where member join is most important, member departure is most important, and where both operations are equally important. We present a stochastic occupancy model that allows one to study the mean behavior of a key tree under different degrees of occupancy. Additionally, we compare the amount of communication overhead

needed in our scheme with the amount of communication overhead that a conventional tree-based rekeying scheme, such as [6], would need to flag users which component of a rekeying message is intended for them.

In Section 4.2 we present an overview of multicast key management for dynamic groups. In Section 4.3 we introduce a method for distributing keys using polynomial interpolation and parametric one-way functions. This basic scheme is used as a building block for a protocol primitive described later in the chapter. Therefore, we present a study of its security and communication features. In Section 4.4 we present some protocol primitives and use these to construct more complex key management operations capable of maintaining the key hierarchy in scenarios with dynamic membership. The size of the messages needed for updating the keys is computed in Section 4.5 and are used to determine the optimal degree of the key distribution tree. Additionally, we present a comparison between computational requirements of the polynomial interpolation scheme proposed in this chapter and the residue-based scheme described in Chapter 3 and [14, 76].

4.2 Review of Multicast Key Management

In this section, we provide a review of key management techniques. Some of the material presented here is repeated from Chapter 3 and is presented for ease of reference.

The distribution of identical data to multiple parties using the conventional point-to-point communication paradigm makes inefficient usage of resources [77]. The redundancy in the copies of the data can be exploited in multicast communication by forming a group consisting of users who receive identical data, and sending a single message to all group users.

When data is being sent over the network, it is important that only valid members of the multicast group should have access to the data. In order to provide access control to the multicast communication, the data is typically encrypted using a key that is shared by all legitimate group members. The shared key, known as the session encryption key (SK), will change with time, depending on the dynamics of group membership as well as the desired level of data protection.

In order to update the session key, a party responsible for distributing the keys, called the group center (GC), must securely communicate with the users to distribute new key material. The GC shares auxiliary keys, known as key encrypting keys (KEKs), that are used solely for the purpose of updating the session key and other KEKs with group members.

One approach to group key management is provided by the group key management protocol (GKMP) [78]. In this scheme, the GC uses a SK, called a group traffic encrypting key (GTEK) in the GKMP literature, and a group key encrypting key (GKEK). The GC updates the SK by using the GKEK. This allows all group members to be updated using a single encrypted message. A major disadvantage of GKMP, however, is that it is not able to handle member departures, or the compromise of a single member. The compromise of the GKEK means that all future communication is compromised since an adversary can calculate future session keys.

Fiat and Naor [79] present a broadcast key distribution scheme that allows for a single source to transmit a SK to a dynamic subset of privileged users such that no coalition of at most k non-privileged users can acquire the SK. The communication overhead of their scheme is not dependent on the amount of non-privileged members, but instead on the security parameter k and a parameter describing the

probability that a coalition of at most k non-privileged users can acquire the SK.

A common class of multicast key management schemes are the tree-based schemes [6–8], of which the scheme we present in Section 4.4 is an example. These schemes tend to have desirable usage of computation, communication, and storage resources for the user or the group controller. In order to motivate the importance of resource scalability with respect to group the multicast group size n, we discuss a simple key distribution scheme that achieves minimal storage for each user, but has highly inefficient communication complexity.

In the minimal storage scheme where multicast group consists of n users, the group center shares a key encrypting key with each user and all users share the same session key. Upon a member departure, the previous session key is compromised and a new session key must be given to the remaining group members. To distribute the new SK, the GC encrypts the new session key with each user's key encrypting key and sends the result to that user. Thus, there are $n-1$ encryptions that must be performed, and $n-1$ messages that must be sent on the network. The storage requirement for each user is 2 keys while the GC must store $n+1$ keys. This approach to key distribution has linear communication, computation and GC storage complexity. As n becomes large these complexity parameters make this scheme undesirable.

The problem of designing efficient key updating schemes is has seen recent attention in the literature. One approach for achieving scalability is to apply hierarchical subgroups and map the KEKs to a logical tree. The tree-based approach to group rekeying was originally presented in [7], and independently in [6]. Due to the tree structure, the communication overhead is $\mathcal{O}(\log n)$, while the storage for the center is $\mathcal{O}(n)$ and for the receiver is $\mathcal{O}(\log n)$. The $\mathcal{O}$ notation is used to indi-

cate that the constant factors are implementation dependent. In [55], it was shown that the optimal tree-based key distribution for a group leads to Huffman trees and the average number of keys assigned to a member is related to the entropy of the statistics of the member deletion event.

Various modifications to the tree scheme have been proposed. In [8], a modification to the scheme of [7] is presented. By using pseudo-random generators, their scheme reduces the usage of communication resources by a factor of two. Similarly, the communication requirements were reduced in [54] by using one-way function trees. In [49] the tradeoffs between storage and communication requirements are studied, and a modification to the schemes of [7] and [6] is presented that achieve sublinear storage. Their results were further explored by Poovendran in [80], where the storage-communication tradeoff is formulated as a convex optimization problem.

In the previous chapter, it was proposed to use multicast key management schemes in conjunction with data embedding to provide and maintain access control to multimedia streams [14, 65, 76]. In this work, a tree-structured multicast key management scheme was proposed that uses residue operations and one-way functions to update keys during member join and member departure operations.

4.3 Basic Polynomial Interpolation Scheme

In this section we describe the basic scheme for distributing keys that will be used in the scalable key management protocol of Section 4.4. The basic key distribution scheme that we describe is a modification of the polynomial interpolation scheme of [52]. We have introduced the use of one-way functions and a broadcast seed to protect private user KEKs from compromise and allow private user KEKs to be

reused.

We shall use parametric one-way functions in our work to provide computational security. A one-way function h is a function from $\mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{Z}$ such that given $z = h(x, y)$ and y it is computationally difficult to determine x [69]. Parametric one-way functions (POWF) are families of one-way functions that are parameterized by the parameter y. Symmetric block ciphers can be used to construct POWFs. Let $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, and consider a symmetric cipher $E_x(y) : \mathcal{Y} \rightarrow \mathcal{Y}$ where the subscript denotes the key used in the encryption of the plaintext y. Thus $\mathcal{X}$ is the key space of the cipher E, while $\mathcal{Y}$ is the space of plaintexts and ciphertexts. Define a hash function $f : \mathcal{Y} \rightarrow \mathcal{Z}$. Then the function $h(x, y) = f(E_x(y))$ is a POWF parameterized by y since any reasonable cryptosystem can withstand a known-plaintext attack, that is knowledge of $E_x(y)$ and y does not make it easy to determine the key x. Note that it is not necessary that the hash function f have any cryptographic properties as the required cryptographic strength is provided by E. Throughout this chapter we shall assume the existence of parametric one-way functions that map sequences of $2B$ bits into sequences of B bits.

Consider the basic key distribution scheme depicted in Figure 4.1. Each user u_i has a personal B-bit KEK K_i that is known only by the group center and user u_i. Additionally, all of the users share a B-bit root KEK $K_\epsilon(t)$ and a session key $K_s(t)$ that will vary with time t.

Suppose that user u_n decides to depart, then we must renew the keys $K_\epsilon(t-1)$ and $K_s(t-1)$ since they were shared by u_n and the other users. The first step is to send the new $K_\epsilon(t)$ to the remaining users. In the polynomial scheme, each user u_i has the distinct pair $(z_i, K_i) \in Z_p \times Z_p$, where Z_p denotes the integers modulo the prime p. The z_j are public knowledge, and are not considered as part of the

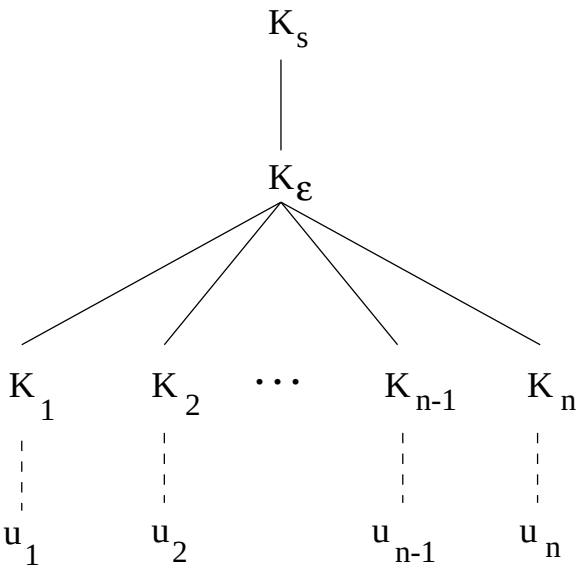


Figure 4.1: The basic key distribution scheme used in the polynomial interpolation method.

secret information that the user must store. Instead, the z_j is any quantity that is used to identify the user, for example a processor id. The GC has made available f, a POWF taking $2B$ bits to B bits. The GC first broadcasts the seed $\mu(t)$ to everyone. Next, the GC associates the following quantity with each user u_j

$$w_j = K_\epsilon(t) + f(K_j, \mu(t)) \pmod{p}. \tag{4.1}$$

The GC generates a degree $n-2$ polynomial $p(z)$ that interpolates the points (z_j, w_j), i.e. $p(z_j) = w_j$. The GC represents $p(z)$ as

$$p(z) = \sum_{i=0}^{n-2} c_i z^i \pmod{p} \tag{4.2}$$

and transmits the message $\alpha_\epsilon(t) = (c_0, c_1, \cdots, c_{n-2})$ to update $K_\epsilon(t)$. This completes the action needed by the GC to update the root KEK, and the session key is then updated using $K_\epsilon(t)$ by transmitting $\alpha_s(t) = E_{K_\epsilon(t)}(K_s(t))$.

A member u_j can calculate $p(z_j) = w_j$ and $f(K_j, \mu(t))$, and hence can recover

$K_\epsilon(t)$.

4.3.1 Resistance to Attack

There are two sources of adversaries for a key management scheme. The first type of adversary is an *external* adversary. This type of adversary is not a member of the service, but receives the encrypted content as well as the rekeying messages. In order for the external adversary to cheat the service, he must mount a successful attack against the rekeying messages in order to acquire the session key, which is needed to decrypt the content. The second type of adversary is an *internal* adversary, who is a member that uses the rekeying messages and his knowledge of his keys to attempt to acquire another user's keys. If an internal adversary can successfully acquire another user's keys, he may cancel his membership to the service, and use the compromised keys belonging to another user to enjoy the service without having to pay.

In the polynomial scheme, an external adversary receives α_ϵ as well as $\alpha_s(t)$. In order for the adversary to acquire the SK, he must mount a successful attack against the cipher used in forming the message $\alpha_s(t)$. Careful selection of a strong cipher algorithm that has received serious study, such as Rijndael [81], will make a successful attack of the SK rekeying message unlikely. Even should a successful attack of the SK rekeying message take place, a future update of the SK would require a subsequent successful attack of the SK rekeying message, which is equally unlikely. Hence, a successful attack against the SK rekeying message would only be a short-lived victory for a pirate.

A second method for acquiring the session key is to attack the message α_ϵ. Given the message $\alpha_\epsilon(t)$, and knowledge of a z_j, it is possible that an adversary

may calculate w_j. However, the adversary must either determine $K_\epsilon(t)$ or a user's $f(K_j, \mu(t))$ given $w_j = K_\epsilon(t) + f(K_j, \mu(t)) \pmod{p}$. The modulo operation makes w_j independent of either $K_\epsilon(t)$ or $f(K_j, \mu(t))$. Should an external adversary successfully attack $K_\epsilon(t)$, then he may acquire the session key. However, upon the next update of the session key, he must make another successful attack upon the root KEK.

The only method for an external adversary to be able to repeatedly acquire the SK is to mount a successful attack on a user's personal key K_j. This requires successful determination of $f(K_j, \mu(t))$ given w_j, which requires searching a space of order p possibilities, and then successfully attacking the one-way function to acquire K_j. The strength of the one-way function should be as strong as the strength of the encryption used to protect the SK rekeying message.

We now discuss the susceptibility of the original polynomial scheme of [52] to *internal* attacks. In the discussion that follows, we refer the reader to Section 3.1 of [52]. For simplicity, we shall assume that the same key K is being distributed to all of the users. Observe that since the z_j-coordinates are public knowledge, an internal adversary may calculate w_j by evaluating the interpolating polynomial at z_j. With knowledge of w_j, the adversary may use his knowledge of K to determine user u_j's private information. Thus, the polynomial scheme of [52] does not protect the private information of each user, and hence cannot be used more than once. If both the z_j coordinate and the personal key K_j are kept secret, then an adversary's task is to search Z_p for any of the n user's z_j coordinate. This is more difficult for an adversary to attack, but also requires both the server and the clients to store twice as much secret information.

As we shall describe in Section 4.5.3, we chose to pursue a different approach

to ensuring the sanctity of each user's private information in order to reduce the communication overhead in our protocol. An inside adversary u_i who desires to calculate another user's key information K_j can calculate $p(z_j) = w_j$, and therefore can calculate $f(K_j, \mu(t)) = w_j - K_\epsilon(t) \pmod{p}$. However, it is difficult for him/her to calculate K_j given $\mu(t)$ and $f(K_j, \mu(t))$ since f is a parametric one-way function. Additionally, should two or more users collude, their shared information does not provide any advantage in acquiring another user's K_j.

4.3.2 Anonymity Reduces Communication Overhead

The above scheme is used in constructing a protocol primitive in the following section. In the protocol primitive, there is a parent key K_ϵ and a handful of sibling keys K_j that are used to update the parent key. Unlike the example described above, application of the protocol primitive might not use all of the sibling keys to update the parent key. This scenario might occur when the GC knows that a sibling key has become compromised or invalidated.

Suppose that there are a possible sibling keys and that m of those sibling keys are used to update the parent key. In a conventional key distribution scheme, such as [6], the update to the parent key is performed by a rekeying message of the form

$$\alpha = \{E_{K_{j_1}}(K_\epsilon) \| E_{K_{j_2}}(K_\epsilon) \| \cdots \| E_{K_{j_m}}(K_\epsilon)\} \tag{4.3}$$

where j_k denotes the sequence representing the m sibling keys used in updating parent key, and $\|$ denotes message concatenation. In addition to the rekeying message, it is necessary to transmit the amount m of children keys, and the user ID message $\{j_1, j_2, \cdots, j_m\}$, which specifies which portion of the rekeying message a user needs in order to determine the new session key.

The transmission of the user ID message in the conventional scheme reveals which sibling keys are still valid. However, it requires that $\lceil \log_2 a \rceil$ bits to represent m and $m\lceil \log_2 a \rceil$ bits to represent $\{j_1, j_2, \cdots, j_m\}$. The total communication overhead of the conventional scheme is thus $(m+1)\lceil \log_2 a \rceil$ bits.

The polynomial interpolation scheme creates a composite message that does not require any user ID message, but instead requires the broadcast of the seed $\mu(t)$. The polynomial scheme defines the rekeying message as the output of a function $PolyInt$ which returns the coefficients of the interpolating polynomial, thus

$$\alpha = PolyInt(K, \{z_{j_1}, z_{j_2}, \cdots, z_{j_m}\}, \{K_{j_1}, K_{j_2}, \cdots, K_{j_m}\}, \mu(t)). \tag{4.4}$$

The input to $PolyInt$ is the key K that is to be distributed, the set of valid non-secret ID parameters $\{z_{j_1}, z_{j_2}, \cdots, z_{j_m}\}$, the broadcast seed $\mu(t)$, and the set of valid sibling keys $\{K_{j_1}, K_{j_2}, \cdots, K_{j_m}\}$. Given a valid sibling key and the seed $\mu(t)$, the new parent key can be determined. On the other hand, an invalid sibling key is unable to determine the new parent key.

If the prime p used in the polynomial scheme has the same bit length as the output of one of the encryptions E_K, then the message size of the polynomial scheme will be the same as the rekeying message of the conventional scheme. If B_μ is the bit length of the broadcast seed, then a measure of comparison between the conventional scheme and the polynomial scheme is the difference $(m+1)\lceil \log_2 a \rceil - B_\mu$. For a single sibling update of the parent node, this difference might favor the conventional approach. The advantage of the polynomial scheme becomes more pronounced when used in a multi-level tree as in Section 4.4. We shall discuss this further in Section 4.5.

4.4 A Scalable Protocol

In the previous section we described the basic scheme for distributing keys during member departures. The basic polynomial interpolation scheme had linear communication requirements during member departures. We now describe a scalable protocol that provides renewal of security levels, handles membership changes, provides a mechanism for reinserting valid members, and allows for the transferal of access rights.

In order to achieve improved scalability, we use a tree-based key hierarchy as depicted in Figure 4.2. In general, the tree can be an a-degree tree. Attached to the tree above the root node is the session key K_s. Each node of the tree is assigned a KEK which is indexed by the path leading to itself. Additionally, each node has a non-secret ID variable z_σ which is used as a non-secret parameter for the $PolyInt$ function. The symbol ϵ is used to denote the root node. Each user is assigned to a leaf of the tree and is given the KEKs of the nodes from the leaf to the root node. Additionally, all users share the session key K_s. For example, user u_{111} is given the keys K_{111}, K_{11}, K_1, K_ϵ, and K_s.

In the protocol that follows, the GC transmits messages to the users via a broadcast channel. It is assumed that each user has an *upstream* channel with minimal bandwidth that is available to convey messages to the GC, such as informing the GC of the intent to depart the service.

The messages that the GC broadcasts to the users must have a standardized structure that is known to all receivers. There are two basic message formats as depicted in Figure 4.3. The first contains three components while the second has five components. The function $B()$ is used to denote the bit length of its operand, thus $B(\sigma)$ is the amount of bits needed to represent σ. The variable Operation

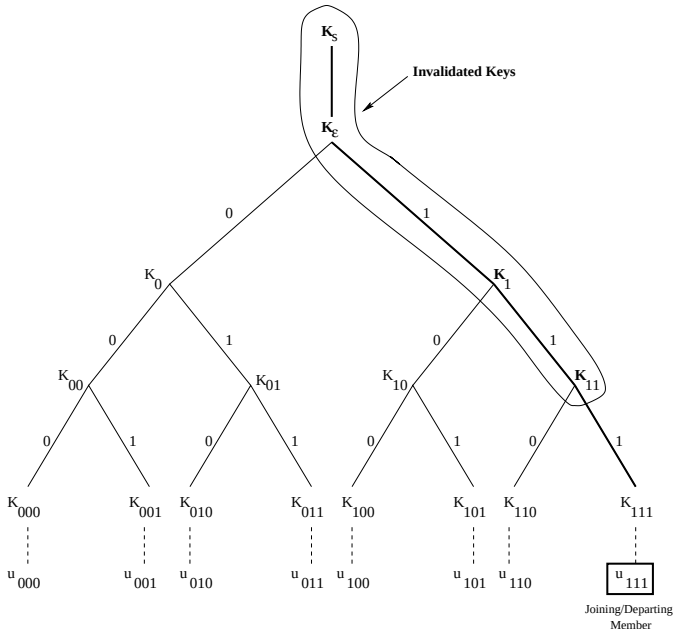


Figure 4.2: Tree-based key distribution.

ID flags the user which protocol primitive is about to be performed. Only five primitive operations are used, and we may therefore represent Operation ID using a 3 bit string. Table 4.1 maps the primitive operations with their corresponding ID bit string.

In the work that follows, we assume that the tree has degree a, and that there are L levels to the tree. The amount of multicast group members n is limited by the amount of leaf nodes on the tree. Thus $n \leq a^L$.

4.4.1 Basic Protocol Primitives

We have identified five basic operations needed in building a system that allows for the update and renewal of the key hierarchy. We now describe each case.

1. Primitive-1(Update SK): This basic operation uses the current root KEK

bit ID	primitive
000	Primitive-1
001	Primitive-2
010	Primitive-3
011	Primitive-4
100	Primitive-5

Table 4.1: Mapping between primitive operations and their corresponding ID bit string.

K_ϵ to update the session key via the rekeying message

$$\alpha = E_{K_\epsilon(t)}[K_s(t)] \tag{4.5}$$

The message format is depicted in Figure 4.3(a). We assume that the maximum size that α can be is 256 bits, and we therefore need 8 bits to represent $B(\alpha)$. This choice of bit length for α would allow for the use of encryption algorithms with a key size of up to 256 bits.

2. Primitive-2(Transmit Seed): The broadcast seed is used in the polynomial scheme to provide protection of secret information. Additionally, it plays a role in reducing the communication overhead associated with flagging the users which part of the message is intended for them. The broadcast of the seed $\mu(t)$ does not require encryption to protect it. The message format for the transmission of the broadcast seed is depicted in Figure 4.3(a). Here $\alpha = \mu(t)$, and $B(\alpha)$ is the amount of bits needed to represent $\mu(t)$. Again, we assume that the maximum size of α is 256 bits, and that 8 bits are used to represent $B(\alpha)$.

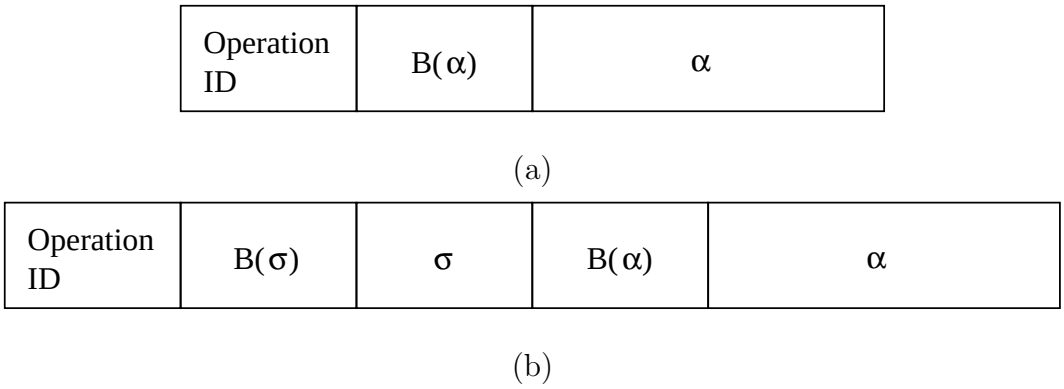


Figure 4.3: The two message structures used in the protocol primitives.

3. Primitive-3(Self Update): It is often necessary for a node, indexed by the a-ary symbol σ, to have its associated key updated using the key at the previous time instant. Thus we will go from $K_\sigma(t-1)$ to $K_\sigma(t)$ by the following message

$$\alpha = E_{K_\sigma(t-1)}[K_\sigma(t)]. \tag{4.6}$$

In this case, we need to flag the receivers which node is being updated. This requires the transmission of the a-ary representation of the node, as well as the amount of bits needed to represent the node. This is depicted in Figure 4.3(b) by the $B(\sigma)$ and σ components of the message. The rest of the message contains the bit length of the message α and the actual rekeying message α. Since the maximum depth of the tree that needs to be represented is $L-1$ and the tree is an a degree tree, the maximum amount of bits needed to represent σ is $\lceil \log_2 a \rceil (L-1)+1$, where the addition of 1 bit was included to account for the need to represent the empty string ϵ as a possible choice for ϵ. In order to represent $B(\sigma)$, we use $\lceil \log_2(\lceil \log_2 a \rceil (L-1)+1) \rceil$ bits. The maximum bit length for α is 256 bits, and 8 bits are used to represent $B(\alpha)$.

4. **Primitive-4(Update Parent):** It is also necessary for the children nodes to update the key of their parent nodes. If σ is the symbol representing the parent node to be updated, then the message

$$\alpha = PolyInt(K_\sigma(t), \{z_{Child(\sigma)}(t)\}, \{K_{Child(\sigma)}(t)\}, \mu(t)) \tag{4.7}$$

is used. Here we have defined the function $Child(\sigma)$ to denote the set of valid children nodes of σ. For example, if we have a binary tree and $\sigma = 00$, and both children nodes are valid, then $Child(\sigma) = \{000, 001\}$. Thus, the message α uses the keys of valid children nodes to update $K_\sigma(t)$. Observe that this message requires that $\mu(t)$ has already been broadcast using Primitive-2, or that the choice of $\mu(t)$ is implicitly known. The message form is depicted in Figure 4.3(b), where again we transfer the bit length of σ and the actual symbol σ to the recipients, followed by the bit length of α and the rekeying message α. We use the same bit allocation for σ and $B(\sigma)$ as in Primitive-3. However, the maximum length for α is aB_{KEK}, and we therefore need $\lceil \log_2 aB_{KEK} \rceil$ bits to represent $B(\alpha)$.

5. **Primitive-5(Reaffirming Parent):** In some operations, it is useful to have a sibling node reaffirm the value of a parent node's key. We define a function $Par(\sigma)$ to denote the symbol corresponding to the parent of the node indexed by σ. To reaffirm the value of a parent node's key, we transmit the message

$$\alpha = E_{K_\sigma(t)}[K_{Par(\sigma)}(t)]. \tag{4.8}$$

The message form is depicted in Figure 4.3(b), and follows the same structure as used in Primitive-3.

4.4.2 Advanced Protocol Operations

We now describe more advanced protocol operations that can be constructed using the primitive operations described earlier. In particular, we focus on the operations of an addition to the membership, a deletion of a user from the membership, the reinsertion of a member into the system, and the transferal of access rights from one user to a new user.

Before we proceed, we present a few comments about how the primitive operations can be used to perform periodic renewal of keying material. Primitive-1 provides a method for performing periodic refreshing of the session key. Refreshing the session key is important in secure communication. As a session key is used, more information is released to an adversary, which increases the chance that a SK will be compromised. Periodic renewal of the session key is required in order to maintain a desired level of content protection, and can localize the effects of a session key compromise to a short period of data. Since the amount of data encrypted using KEKs is usually much smaller than the amount of data encrypted by a session key, it is not necessary to refresh KEKs as often. However, the periodic renewal of a KEK can be performed using Primitive-3.

Member Join

In many applications, such as pay-per-view broadcasts and video conferences, the group membership will be dynamic. It is important to be able to add new members to any group in a manner that does not allow new members to have access to previous data. In a pay-per-view system, this amounts to ensuring that members can only watch what they pay for, while in a corporate video conference there might be sensitive material that is not appropriate for new members to know.

Suppose that a new user contacts the service desiring to become a group member. The new client sends the GC a message detailing the client's credentials, such as identity information, billing information, and public key parameters that the GC may use to communicate with the new client. Mutual authentication between the new client and the GC should be performed. A public key infrastructure, such as X.509 certificates [82], may be used for this purpose. Upon verification of the new user's information, the GC assigns the client to an empty leaf of the key tree. For simplicity of presentation, we assume that the tree has empty slots. If the tree is already full, then the user may either be turned away, or an additional layer must be added to the tree using a separate operation, which is not described in this thesis. The GC then issues the new client his keys via a communication separate from the communications sent to the current group members, as well as informing the new user the time at which those keys will become valid.

Meanwhile, the GC updates the current members of the multicast group. Suppose that the GC plans on inserting the new member into the leaf node indexed by the symbol ω. Then the SK as well as the KEKs on the path from the parent node of ω to the root node ϵ must be renewed. The following algorithm describes how this procedure can be accomplished using the protocol primitives. We use the notation $Par^j(\omega)$ to denote the parent function applied j times to ω. Thus $Par^2(\omega)$ is the *grandparent* of ω.

for $j = 1 : L$ **do**

  $\sigma = Par^j(\omega)$;

  Update $K_\sigma(t-1) \rightarrow K_\sigma(t)$ using Primitive-3 ;

end

Update SK using Primitive-1 ;

Member Departure

Members will also wish to depart the service, and must be prevented from accessing future communication. Assume that user u_ω contacts the GC wishing to depart the service. Upon authenticating the user's identity, the procedure that the GC enacts to remove member u_ω and update the keys of the remaining members is

Generate random $\mu(t)$;

Broadcast $\mu(t)$ using Primitive-2 ;

for $j = 1 : L$ **do**

  $\sigma = Par^j(\omega)$;

  Determine valid children of σ: $Child(\sigma)$;

  Update $K_\sigma(t-1) \rightarrow K_\sigma(t)$ using Primitive-4 ;

end

Update SK using Primitive-1 ;

Member Reinsertion

It might often occur that a valid member, denoted by index ω, misses the rekeying messages needed to update the key hierarchy. The client must notify the GC that he missed rekeying messages using an upstream (client to server) channel. Upon verification of the user's identity, the GC performs the member reinsertion operation, which sends the new user the specific keys he needs to be able to resume the service.

If the service provider has a downstream (server to client) channel available to communicate with the user, then service provider may use this channel to send the needed keys by encrypting them with the user's personal key K_ω. In many

scenarios, however, after the initial contact with the service provider, the client has a low-bandwidth channel for upstream communication, and only the broadcast channel available for downstream communication. In these cases, although only a single user needs the rekeying messages, the rekeying messages must be multicast. Since this user has a valid private key K_ω, the GC can start with this key to provide $K_{Par(\omega)}(t)$ to the user. We can then proceed up the tree, using the sibling key to convey the current status of the parent key. The procedure for this operation is as follows:

for $j = 1 : L$ **do**

  $\sigma = Par^j(\omega)$;

  Convey parent key $K_\sigma(t)$ to siblings using Primitive-5 ;

end

Convey current SK using Primitive-1 ;

An added bonus of using the sibling key to convey the current status of the parent key is that other users may observe these rekeying messages to reaffirm the validity of some of their keys.

Transferal of Rights

Suppose that user u_ω wishes to give his rights to another user who is not currently a member. We will denote this new user by u_{ω_B} to indicate that he will take over the keys on the path from ω to the root node. For the purpose of calculating parent and sibling relationships, ω and ω_B are identical, thus $Par(\omega) = Par(\omega_B)$.

In order to transfer access rights, both users must contact the GC, who performs an authentication procedure to verify that the transferal is legitimate. Then, using a secure channel, the GC gives to user u_{ω_B} its own personal key K_{ω_B}. One method

for creating a secure channel is to use public key cryptography. K_{ω_B} replaces K_ω on the key tree. All of the keys that belonged to u_ω must be changed to prevent u_ω from accessing content that he has given up the right to access. The procedure for transferring access rights is as follows:

Generate random $\mu(t)$;

Broadcast $\mu(t)$ using Primitive-2 ;

for $j = 1 : L$ **do**

  $\sigma = Par^j(\omega_B)$;

  Determine valid children of σ: $Child(\sigma)$;

  Update $K_\sigma(t-1) \rightarrow K_\sigma(t)$ using Primitive-4 ;

end

Update SK using Primitive-1 ;

We observe that the algorithm for transferring rights is nearly identical with the algorithm for removing a member from a group. The difference lies in the fact that user u_{ω_B} is considered a valid user, and hence is a valid child of its parent.

The procedure for user u_ω to reclaim his access privileges is similar. This time, only user u_ω is required to contact the GC requesting that he regain his access privileges. The GC performs an authentication procedure to guarantee that the identity of u_ω is truthful, and then replaces K_{ω_B} with K_ω. The KEKs and SK are changed according to the above algorithm, with ω replacing ω_B.

4.5 Architecture Considerations

4.5.1 Optimization of Tree Degree for Communication

The amount of communication that a rekeying protocol requires affects the speed at which the rekeying scheme can handle membership changes. It is therefore important to minimize the size of the communication used by the key management scheme. In particular, since the two most important operations performed by a multicast key management protocol are membership joins and membership departures, we shall focus on optimizing the tree degree for these two operations.

In what follows, we present a worst-case analysis of the communication requirements for member join and member departure operations. It is observed that member join and member departure operations lead to conflicting optimality criteria. Since a real system will have to cope with both member joins and member departures, we jointly consider the departure and join operations, and present optimization results when both member join and departure operations are equally weighted.

We refer the reader to the protocol descriptions as well as the message structure in Figure 4.3. We shall denote the degree of the tree by a, and the number of levels in the tree by L. B_{SK} shall denote the bit length of session key, B_{KEK} shall denote the bit length of the key encrypting keys, and B_{μ} the bit length of the broadcast seed $\mu(t)$.

Worst-Case Analysis

It is easy to see that, for a given tree, the scenario that produces the most communication for the member join operation occurs when one node on each level from

the root to level $L-1$ must be updated. In this case, all of the KEKs on the path from one user to the root must be refreshed. We now calculate the amount of communication needed to update the tree for this worst-case scenario.

The member join operation consists of two types of operations: updating the KEKs, and updating the SK. In order to update the KEKs, we use Primitive-3 L times. Each step of the loop must send the quintuple (operation ID, bit length of update node $B(\sigma)$, node ID σ, bit length of the update message $B(\alpha)$, update message α). The symbol σ starts near the bottom of the tree, and through application of the Parent function moves toward the root of the tree.

In order to represent the symbol σ during the jth iteration of the loop, we need to convert from base a to base 2 and hence $B(\sigma) = \lceil \log_2 a \rceil (L-j)+1$ bits. In addition, we must send $B(\sigma)$, which requires $\lceil \log_2 (\lceil \log_2 a \rceil (L-1)+1) \rceil$ bits. Here the addition of 1 was to allow for the need to represent the empty string ϵ as a possible choice for σ. Similarly, in each stage of the loop the rekeying message α has bit length $B(\alpha) = B_{KEK}$ and since we have fixed the maximum key length to be 256 bits, we require 8 bits to represent $B(\alpha)$. The update to the session key requires sending the ID flag, $B(\alpha)$ and α. Therefore, the amount of bits needed to update the session key is $3+8+B_{SK}$. The total amount of bits needed to update the key tree during a member join is

$$\begin{aligned} C_{MJ} &= \left(\sum_{j=1}^{L} \Big[3 + \lceil \log_2 (\lceil \log_2 a \rceil (L-1)+1) \rceil + \lceil \log_2 a \rceil (j-1) + 9 + B_{KEK} \Big] \right) \\ &\quad +3+8+B_{SK}. \end{aligned}$$

The amount of communication needed in the member departure case can be similarly calculated. The main difference between member join and member departure is that there are three operations: the broadcasting of $\mu(t)$, updating the KEKs, and updating the SK. The most communication occurs when $a-1$ nodes

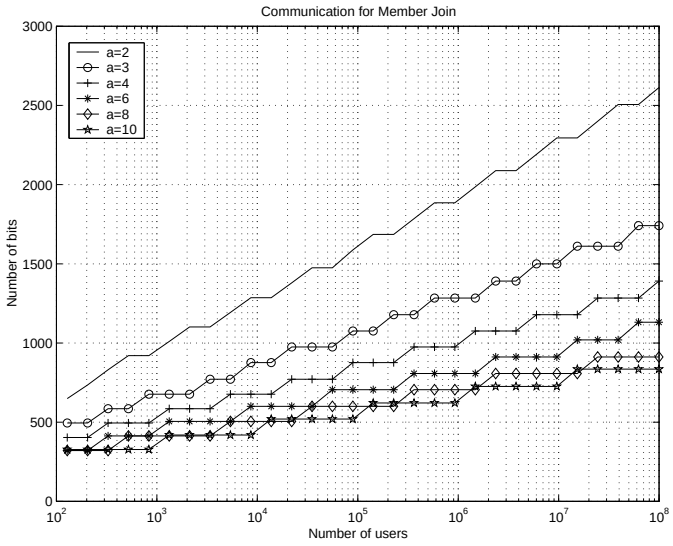
on level L must be used to update the key on level $L-1$, and a nodes are used to refresh each of the remaining KEKs on the path from the departing member to the root node. After appropriately expanding and gathering terms, the communication for the member departure can be found to be

$$\begin{aligned} C_{MD} &= 22 + B_{SK} + B_{\mu} + (La - 1)B_{KEK} + (L) \\ &\quad \left(4 + \lceil \log_2 aB_{KEK} \rceil + \lceil \log_2 \left(\lceil \log_2 a \rceil (L-1) + 1 \right) \rceil + \frac{(L-1)}{2} \lceil \log_2 a \rceil \right). \end{aligned}$$

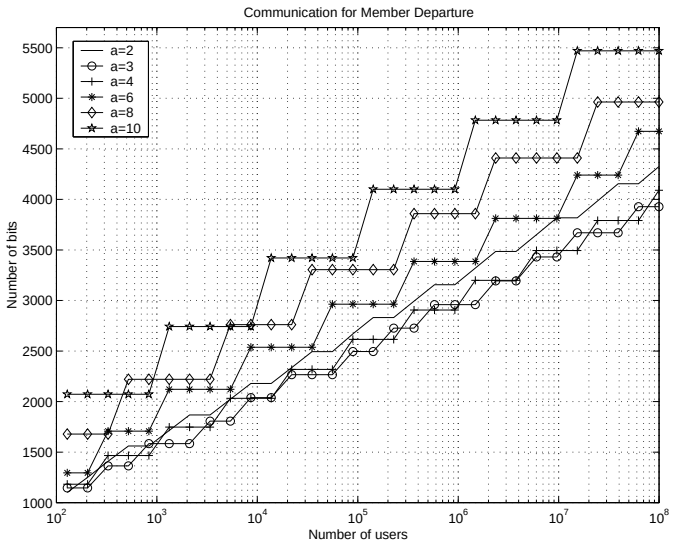
We calculated the worst-case amount of communication required to update an a-degree key tree as a function of the number of users n with the amount of tree levels set to $L = \lceil \log_a n \rceil$. In our calculations, we chose $B_{SK} = B_{KEK} = B_{\mu} = 64$ bits. We chose to use 64 bits as the key size since such a key length can provide strong levels of security when used with some ciphers, such as RC5 [69]. The amount of communication required for different choices of the degree of the tree a during a member join is depicted in Figure 4.4(a). This figure shows the general trend that less communication is required during member join operations if we use a higher degree tree. On the other hand, Figure 4.4(b) shows the amount of communication needed during the worst case of a member departure operation. In this case, the larger tree degrees are definitely not advantageous. It is also evident that a binary tree is not optimal when considering member departure. In fact, the values of $a = 3$ and $a = 4$ appear to be the best choice, with optimal choice fluctuating depending on n.

Joint Departure-Join Optimization

In some application scenarios the key tree might start out relatively empty, and the amount of member join operations would be greater than the amount of member departure operations. In this case, the membership grows towards the tree



(a)



(b)

Figure 4.4: (a) The amount of communication C_{MJ} required during member join operations for different tree degrees a and different amounts of users n. (b) The worst case amount of communication C_{MD} required during member departure operations for different tree degrees a and different amounts of users n.

capacity, and the communication required for the member join operation is more critical than the communication for member departure. On the other hand, some scenarios might start out with a nearly full key tree, and the member departure operation would outweigh the member join operation.

We therefore would like a communication measure that runs the gamut between the two extremes of just the member join communication, and just the member departure communication. This can be accomplished by considering the convex combination of C_{MJ} and C_{MD}.

Let λ denote the probability of a member departure operation, and assume that $1 - \lambda$ is the probability of a member join operation, then the combined communication measure C_C given by

$$C_C = \lambda C_{MD} + (1 - \lambda) C_{MJ} \tag{4.9}$$

weights the member departure and member join operations according to their likelihood. For example, when $\lambda = 0$ the emphasis is entirely placed on the member join operation, while $\lambda = 1$ corresponds to when the emphasis entirely placed on the member departure operation. The case of $\lambda = 0.5$ corresponds to equal emphasis on the two operations, which is depicted in Figure 4.5. From this figure, we see that the choice of $a = 4$ stands out as the best choice for $n > 10000$ when equally weighting the member join and member departure operation.

4.5.2 Binomial Occupancy Model

Since it is very difficult to calculate the amount of communication needed during membership changes when a specific amount of users n are placed on the tree, we have devised a stochastic model that allows one to study the behavior of the system when there are varying amounts of occupancy. We assume that the leaf nodes of

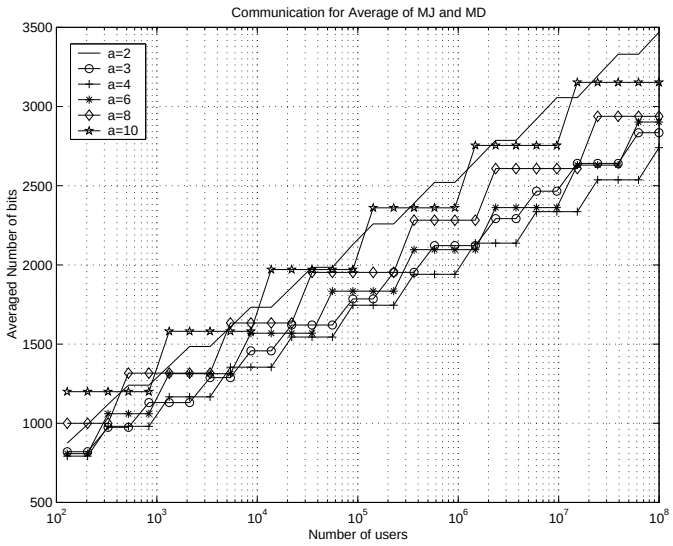


Figure 4.5: The average of C_{MD} and C_{MJ} for different tree degrees a and different amounts of users n.

the a-degree key tree with L levels are occupied according to i.i.d. Bernoulli distributions with a probability of occupancy q_L. This implies that the occupancy n is modeled according to a binomial distribution with mean occupancy $q_L a^L$ and variance $q_L(1-q_L)a^L$. Hence, when q_L is higher, the tree is on average at higher occupancy.

We first calculate the average amount of communication required for member join when the probability of a node being occupied is q_L. Let τ_a denote the a-ary representation of the joining member. We may denote the siblings of τ_a by $\tau_1, \tau_2, \cdots, \tau_{a-1}$. Define the random variable Z_{L-1} as

$$Z_{L-1} = \begin{cases} 1 & \text{if any } \tau_k \text{ is occupied} \\ 0 & \text{if no } \tau_k \text{ are occupied} \end{cases}.$$

Since the τ_k are occupied with a probability of q_L, we have $P(Z_{L-1}=1) = 1-(1-q_L)^{a-1}$, and the expected value of Z_{L-1} is given by $E(Z_{L-1}) = 1-(1-q_L)^{a-1}$.

We may perform a similar procedure for the other levels. We denote the j-

siblings as those nodes τ such that $Par^j(\tau) = Par^j(\tau_a)$. For level $L-j$, we may define the random variable Z_{L-j} as

$$Z_{L-j} = \begin{cases} 1 & \text{if any } j\text{-sibling node of } \tau_a \text{ is occupied} \\ 0 & \text{if no } j\text{-sibling nodes of } \tau_a \text{ are occupied} \end{cases}.$$

In this case, $P(Z_{L-j} = 1) = 1 - (1 - q_L)^{a^j - 1}$, and the expected value of Z_{L-j} is given by $E(Z_{L-j}) = 1 - (1 - q_L)^{a^j - 1}$.

The average communication requirements for member join can be derived as

$$\begin{aligned} \overline{C_{MJ}} &= \Bigg(\sum_{j=1}^{L}(1-(1-q_L)^{a^j-1})\Big[12 + \lceil \log_2(\lceil \log_2 a \rceil (L-1)+1)\rceil \\ &\quad + \lceil \log_2 a \rceil (L-j) + B_{KEK}\Big]\Bigg) + 11 + B_{SK}. \end{aligned}$$

We now apply the model to calculating the average amount of communication needed during member departure. Again suppose that the departing member is indexed by the a-ary symbol τ_a. Label the siblings of τ_a by $\tau_1, \tau_2, \cdots, \tau_{a-1}$, and define the random variable X_k by

$$X_k = \begin{cases} 1 & \text{if } \tau_k \text{ is occupied} \\ 0 & \text{if } \tau_k \text{ is not occupied} \end{cases}.$$

Let us define $Y_L = \sum_{k=1}^{a-1} X_k$, which is the random variable corresponding to the amount of occupied sibling nodes of τ_a at level L. The probability that i sibling leafs at level L are occupied is given by

$$P(Y_L = i) = \binom{a-1}{i} q_L^i (1-q_L)^{a-1-i}. \tag{4.10}$$

Y_L is thus a binomial random variable with expected value $E(Y_L) = (a-1)q_L$. Hence, the average number of nodes to be updated at level L is $(a-1)q_L$.

At level $L-1$, we know that the parent node of the departing member will automatically be used in updating the next higher level. Since the probability of

a node at level L being occupied is q_L, the probability that a node on level $L-1$, other than $Par(\tau_a)$, being occupied is

$$q_{L-1} = 1 - (1 - q_L)^a \,. \tag{4.11}$$

This time, we may denote the siblings of $Par(\tau_a)$ by $\tau_1, \tau_2, \cdots, \tau_{a-1}$. Again, we define the random variable X_k by

$$X_k = \begin{cases} 1 & \text{if } \tau_k \text{ is occupied} \\ 0 & \text{if } \tau_k \text{ is not occupied} \end{cases} .$$

We now define the random variable Y_{L-1} to be the amount of sibling nodes of $Par(\tau_a)$ that are occupied, and we find that $E(Y_{L-1}) = (a-1)q_{L-1}$. Since we must also include $Par(\tau_a)$ in the updating we must add one. Thus, the expected number of nodes on level $L-1$ that must be updated is $1 + (a-1)q_{L-1}$. We may similarly perform this calculation for level j, where $q_j = 1 - (1 - q_{j+1})^a$, and the expected number of nodes on level j to be updated is $1 + (a-1)q_j$.

In order to calculate the average amount of communication for the member departure operations, we must consider both the expected amount of communication associated with the overhead and the payload of the message. The average communication for the overhead consists of the amount of communication needed to send the operation id, the node id, and the bit length of the update message. This calculation can be done using the expected value of Z_{L-j}. The average communication for the payload is calculated using the expected number of nodes on level j to be updated. The average amount of communication for n users on an a-degree tree with L levels is therefore given by

$$\overline{C_{MD}} \quad = \quad 22 + B_\mu + B_{SK} + q_L(a-1)B_{KEK} + \left(\sum_{j=1}^{L-1} \left(1 + (a-1)q_j\right) B_{KEK} \right)$$

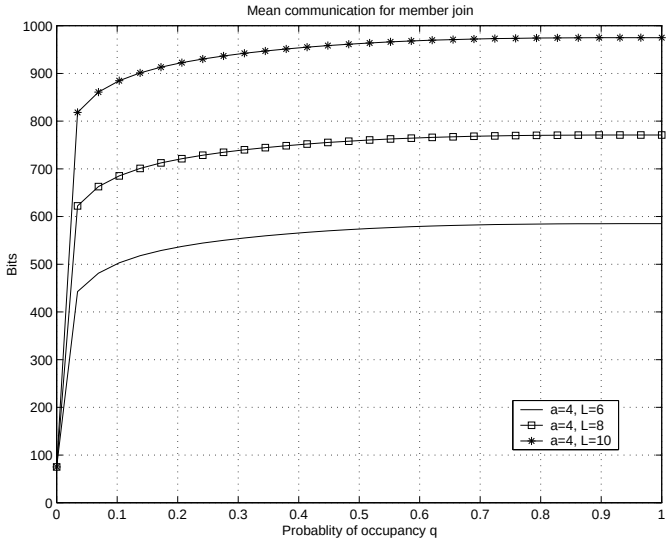
$$+\Bigg(\sum_{j=1}^{L}\Big(1-(1-q_L)^{a^j-1}\Big)\Big(4+\lceil\log_2(\lceil\log_2 a\rceil(L-1)+1)\rceil$$
$$+\lceil\log_2 a\rceil(L-j)+\lceil\log_2 aB_{KEK}\rceil\Big)\Bigg).$$

We calculated the mean message size for member join and member departure operations as parameterized by q when the tree degree is $a = 4$ and there are 6, 8 and 10 levels. The key sizes were chosen to be $B_{SK} = B_{KEK} = B_{\mu} = 64$ bits. In Figure 4.6, we have indicated the mean communication as a function of q. One can see that the expected communication rapidly increases as the probability q becomes slightly greater than 0. In the member join operation, the communication levels off to a flat plateau as the probability of occupancy increases. For the member departure operation, the mean communication also increases rapidly for $q < 0.1$, but then grows less dramatically for higher q. From these two curves, we can infer that a key tree which is roughly half occupied does not have considerably different communication requirements than the worst-case communication requirements, which occur when $q = 1$. This supports our use of the worst-case scenarios for optimizing the tree degree.

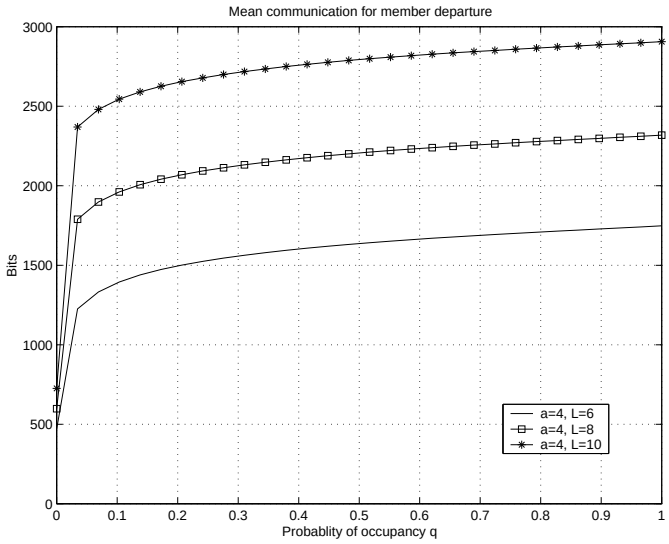
4.5.3 Communication Overhead

Earlier we mentioned that one motivation for using the broadcast seed is that it reduces the amount of communication overhead associated with notifying to the users which rekeying messages are intended for them during member departures. We now explore this concept in the framework of a tree-based scheme.

Consider an a degree tree with n users. In a general tree-based scheme, when a user departs, all of the keys on the path from the departing member's leaf to the root key must be updated. To update a key associated with a particular node σ,



(a)



(b)

Figure 4.6: The expected amount of communication for a degree 4 tree with 6, 8, and 10 levels as a function of the probability q that a leaf node is occupied. (a) Member Join, (b) Member Departure.

we must determine the keys associated with populated children nodes. These keys are then used to encrypt the update, and the rekeying message is then of the form:

$$\alpha = \{E_{K_{j_1}}(K_\sigma) \| E_{K_{j_2}}(K_\sigma) \| \cdots \| E_{K_{j_m}}(K_\sigma)\}. \tag{4.12}$$

Here we have used the sequence $\{j_k\}$ to denote index the symbols of the valid children nodes. In addition to sending the rekeying message, it is necessary to send the number of valid children nodes m, and the sequence $\{j_1, j_2, \cdots, j_m\}$.

The worst case scenario for communication overhead in updating a tree is when a of the children nodes are used to update each parent node. In this case, the communication overhead required is

$$C_O = (a+1)\lceil \log_2 a \rceil \lceil \log_a n \rceil. \tag{4.13}$$

This equation is obtained by considering both the communication needed to send the amount of valid children nodes, and the symbols for each valid child node.

This amount of communication overhead was calculated for different group sizes n and different tree degrees a. The resulting amount overhead is depicted in Figure 4.7. In this figure we have also drawn a baseline corresponding to $B_\mu = 64$ bits, which is the amount of communication overhead required if one uses the Member Departure protocol of Section 4.4. Examining the case of $a = 4$, which corresponds to the optimal value of the tree-degree as previously determined, shows that for values of $n > 10000$, the Member Departure protocol described in this chapter requires less communication overhead in the worst case scenario. Additionally, observe that if we use a higher degree tree, which is better suited to scenarios where more users are joining than departing, then the efficiency of the Member Departure protocol is even more pronounced.

The use of a broadcast seed can gain further improvement if we choose to use

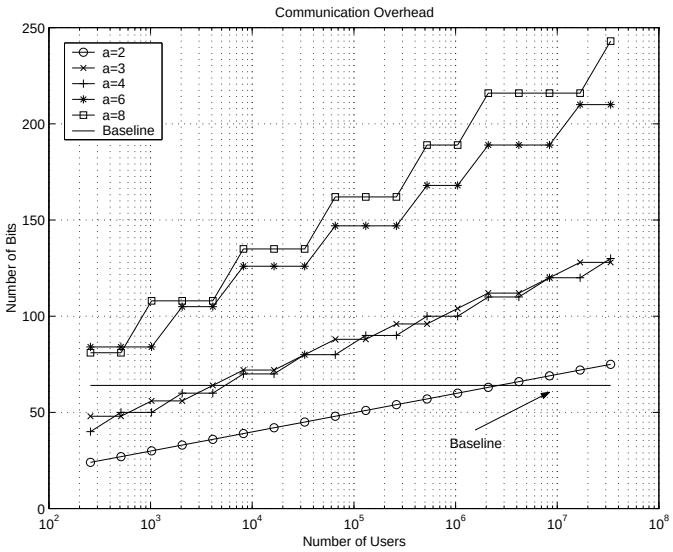


Figure 4.7: The worst-case member departure communication overhead required in a conventional tree-based rekeying for different tree degrees versus the baseline communication required when using the polynomial interpolation scheme. The baseline communication corresponds to $B_\mu = 64$ bits.

$\mu(t) = K_s(t-1)$. In this case, the broadcast seed does not have to be sent since it is known by the remaining users. Therefore, there is no communication overhead associated with updating during member departure, and we may consider the baseline at $B_\mu = 0$. In this case, the benefits of using a broadcast scheme becomes even more pronounced.

4.5.4 Computational Complexity

We have seen that one advantage of broadcast schemes is that they reduce the amount of communication overhead associated with sending flagging messages. It should be apparent that a message form like equation (4.3) takes less computation to form than a message form like equation (4.4) assuming that calculating $E_K(K_\sigma)$

has comparable computation as $f(K_\sigma, \mu(t))$. Hence, to rekey using our message form requires more computation than when using a conventional rekeying message structure.

We now compare the worst-case computational complexity of the rekeying form of this chapter with the residue-based rekeying form proposed in [76]. We shall only focus on the update of the KEKs during a member departure since the other operations are identical.

In the residue-based rekeying form, each level of the key tree is rekeyed by calculating the product of a numbers, each requiring $B+1$ bits to represent, and the addition of a B bit number with a $(B+1)^a$ bit number. Using the fact that adding a k bit number to an l bit number requires $max(k,l)$ bit operations, and multiplying a k bit number by an l bit number requires $\mathcal{O}(kl)$ bit operations [83], we therefore find that for an L level tree, the computational complexity of the residue-based message form is $\mathcal{O}(LB^a)$.

Similarly, in the scheme of this paper, we have L levels of KEKs to update. At each level of the tree we must calculate the coefficients of a degree $a-1$ interpolating polynomial, except at the bottom level where we must calculate the coefficients of a degree $a-2$ polynomial.

In order to calculate the coefficients of a s-degree interpolating polynomial, we use the Newton form of the interpolating polynomial [84]. Algorithm 5 is a modification of the polynomial interpolation algorithm of [85], that can be used to determine the coefficients β_j of the s-degree polynomial that interpolates the points $(z_j, g_j) \in Z_p \times Z_p$, where $j \in \{0, 1, \cdots, s\}$. The algorithm writes the β_j values into the input array values g_j.

This algorithm requires addition, multiplication, inversion, and modulo oper-

for *k=0:s-1* **do**

  for *j=s:-1:k+1* **do**

    $g(j) = (g(j) - g(j-1))(z(j) - z(j-k-1))^{-1} \pmod{p}$

  end

end

for *k=s-1:-1:0* **do**

  for *j=k:s-1* **do**

    $g(j) = g(j) - g(j+1)z(k) \pmod{p}$

  end

end

Algorithm 5: Algorithm for determining the coefficients of an interpolating polynomial.

ations to take place modulo p. The most intensive operation of these is that of inverting a number. Assume that the prime p is chosen to have B bits, then the amount of bits operations needed to calculate the inverse of a number modulo p using the Euclidean algorithm is $\mathcal{O}(B^3)$ [83]. The above algorithm requires $\frac{s(s+1)}{2}$ inversions in order to determine a degree s interpolating polynomial. Therefore, the amount of bit operations needed to update an L level degree a key tree using the polynomial interpolation scheme is $\mathcal{O}(a^2LB^3)$.

Comparing the two computational estimates $\mathcal{O}(LB^a)$ and $\mathcal{O}(a^2LB^3)$ indicates that for higher tree degrees a, the scheme based upon polynomial interpolation asymptotically requires less bit operations and is more computationally efficient.

4.6 Chapter Summary

In order to address the problem of managing keys for securing multicasts, we proposed a framework that is suitable for dynamic group environments. Advanced protocol operations that update the keys during member joins, member departures, and the transferal of access rights were built using basic protocol operations which we call protocol primitives.

We described several desirable features for a multicast key management scheme, and which our scheme satisfied. In particular, our architecture provides a method for renewing session keys and key encrypting keys needed to control access to content. By using either the basic protocol operations, or more advanced protocol operations, the session key or key encrypting keys can be refreshed when a key's lifetime expires due to age or changes in membership. It is also evident that if users were to collude, they would not be able to figure out keys that they did not have. Users may survive accidents or move across terminals by sending a request for reinsertion to the server, upon which the server performs the member reinsertion protocol operation. We also provided a description of a protocol operation that would allow users to transfer their access rights to other parties. The server can revoke access to an individual by using the member departure operation to remove the member from the key hierarchy. Finally, our protocol uses a tree-structured key hierarchy in order to achieve desirable communication requirements during changes in the group membership.

A novel feature of this scheme is that it uses polynomial interpolation in conjunction with a broadcast seed to handle member departure operations. We studied the communication associated with performing member join and member departure operations. It was observed that higher tree degrees are best for member join

operations, whereas a tree degree of 3 or 4 was best for the member departure operation. When equally weighting the join and depart operations, a degree 4 tree stood out as optimal. The communication overhead of the polynomial interpolation scheme is reduced in comparison to a model conventional scheme. We provided a comparison between the communication overhead of our scheme and the overhead of an example conventional scheme that used ID messages to flag the users which parts of the rekeying message were intended for them. As group size and tree degree increased, the communication overhead for the conventional scheme increases and ultimately becomes more burdensome than sending the broadcast seed. For example, when the group size was $n = 100000$ and the tree degree was $a = 4$, the communication overhead in the conventional scheme was approximately 25 % more than the overhead associated with a broadcast seed of size $B_\mu = 64$ bits. Finally, if one uses the previous session key $K_s(t-1)$ as the seed $\mu(t)$, then no communication overhead is associated with our protocol during member departures.

We presented a study of the communication needed when using our architecture to perform member joins and member departures. These two operations are the most important operations that a multicast server will have to face when operating in dynamic environments. The communication requirements of the member join and member departure operations lead to conflicting tree design considerations. By explicitly computing these two quantities as functions of the degree of the tree and computing the communication overheads, we studied the tree selection criterion. From our computations, the communication during a member join is reduced when using a higher degree tree, while the optimal tree degree for a member departure is either $a = 3$ or $a = 4$. We considered the average of the communications for the two operations, which gave strong support to choosing $a = 4$ as the optimal tree

degree. We presented a stochastic population model that allows one to study the mean behavior of our architecture for varying amounts of users. It is observed that for both the join and departure operation, the amount of communication needed to update the key tree rapidly increases as the tree approaches 10% population. Above 10% occupancy, the communication needed for both operation stabilizes. We also compared the computational requirements of the tree-based rekeying schemes using polynomial interpolation and residue arithmetic. Estimates of the amount of bit operations needed in both cases indicates that the polynomial interpolation scheme requires less computation for higher degree trees.

We have now completed the discussion of new security techniques to establish and maintain the information needed to provide access control to group communications. In the next chapter, we shall shift the focus to addressing the important security issue of controlling content usage and redistribution after the data leaves the protection of access control technologies.

Chapter 5

Anti-Collusion Fingerprinting for Multimedia

5.1 Introduction

The advancement of multimedia technologies, coupled with the development of an infrastructure of ubiquitous broadband communication networks, promises to facilitate the development of a digital marketplace where a broad range of multimedia content, such as images, video, audio and speech, will be available. However, such an advantage also posts the challenging task of insuring that content is appropriately used. Before viable businesses can be established to market content on these networks, mechanisms must be in place to ensure that content is used for its intended purpose, and by legitimate users who have purchased appropriate distribution rights.

Although access control is an essential element to ensuring that content is used by its intended recipients, it is not sufficient for protecting the value of the content. The protection provided by encryption disappears when the content is no longer in the protected domain. Whether the content is stored in an unencrypted format, or decrypted prior to rendering, it is feasible for users to access cleartext representations of the content. Users can then redistribute unencrypted representations,

which affects the digital rights of the original media distributors.

In order to control the redistribution of content, digital fingerprinting is used to trace the consumers who use their content for unintended purposes [10, 11, 86]. These fingerprints can be embedded in multimedia content through a variety of watermarking techniques [12, 13]. Conventional watermarking techniques are concerned with robustness against a variety of attacks such as filtering, but do not always address robustness against a coalition of users with the same content that contains different marks. These attacks, known as *collusion* attacks, can provide a cost-effective approach to removing an identifying watermark. One of the simplest approaches to performing a collusion attack is to average multiple copies of the content together [87]. Other collusion attacks might involve forming a new content by selecting different pixels or blocks from the different colluders' content. By gathering a large enough coalition of colluders, it is possible to sufficiently attenuate each of the colluders' identifying fingerprints and produce a new version of the content with no detectable fingerprints. It is therefore of ample importance to design fingerprints that are not only able to resist collusion, but also identify the colluders. Such a scheme also provides a means to discourage attempts at collusion by the users.

The problem of designing fingerprints that are resistant to collusion has been considered for generic digital data in [10, 11]. Such generic schemes, however, do not consider the actual marking process associated with specific applications and media types. Indeed, the design of collusion resistant fingerprinting should consider application-specific issues such as the inherent, special properties of multimedia data since the fingerprinting process for multimedia involves a chain of events including the selection of the embedding method, appropriate choice of detection

statistics, and the application environments.

In this chapter, we investigate the problem of making fingerprints for multimedia content, such as images and video, that are resistant to collusion attacks by averaging. We show that under reasonable assumptions, the optimal fair strategy for a group of colluders performing an averaging collusion attack is to perform an average where each user weighs their marked content equally. We then study the effect that collusion has upon the constellation points in orthogonal and simplex modulation. In order to overcome the linear complexity associated with traditional detection schemes for orthogonal modulation, we develop a tree-based detection scheme that is able to efficiently identify K colluders with an amount of correlations that is logarithmic in the number of basis vectors. A drawback of orthogonal modulation for embedding is that it requires as many orthogonal signals as users. Since many applications will desire to distribute content to vast quantities of consumers, it is desirable to squeeze more users into fewer signals. We propose to exploit the important interplay between the coding, embedding and detection process to design a family of anti-collusion codes (ACC) that are appropriate for different multimedia scenarios and can accommodate more users with fewer orthogonal signals than in orthogonal modulation. The purpose of ACC is not only to resist collusion, but also to trace who the colluders are. The proposed ACC are used with code modulation to fingerprint multimedia, and have the property that the composition of any subset of K or fewer codevectors is unique, which allows us to identify groups of K or fewer colluders. We present a construction of binary-valued ACC under the logical AND operation that uses the theory of combinatorial designs and is suitable for both the on-off keying and antipodal form of binary code modulation. Our code construction is able to accommodate n users, while requir-

ing only $\mathcal{O}(\sqrt{n})$ basis vectors. This is a reduction in the amount of signals needed when using orthogonal modulation for embedding fingerprints. Our approach is suitable for both averaging-based collusion attacks, and for collusion attacks that interleave values or pixels from differently marked versions of the same content. To demonstrate the concept of ACC and fingerprinting, we will focus on additive embedding with correlation-type detection. The proposed ACC concept is applicable to all multimedia data types. For the convenience of discussion, we will use images as an example, while the extension to audio or video is quite straightforward.

This chapter is organized as follows: In Section 5.2 we describe multimedia fingerprinting, and introduce the problem of user collusion for a class of additive watermark schemes. We then review orthogonal modulation in Section 5.3, and describe the effect that collusion has upon the constellation points of the modulation scheme. We describe an efficient detection scheme, and present simulations to demonstrate the behavior of our efficient detector in a collusion scenario. In Section 5.4, we present our design of anti-collusion codes, which are used in conjunction with binary code modulation to accommodate more users for a given amount of orthogonal signals. Our ACCs are based upon assumptions about the behavior of the detector, and we therefore study the detector and present simulations for both an abstract model consisting of Gaussian signals, and real images. Finally, we present conclusions in Section 5.5.

5.2 Fingerprinting and Collusion

We begin by introducing the techniques of additive data embedding, which may be used to embed fingerprints into multimedia sources. The problem of embedding fingerprints is closely married to the problem of detecting the fingerprints, and

we therefore provide a brief review of the elements of detection theory that are relevant to the problem. Later in this section we will introduce the averaging-based collusion attack.

5.2.1 Fingerprint Detection

In this section, we will review additive embedding, where a watermark signal is added to a host signal. Suppose that the host signal is a vector denoted as $\mathbf{x}$ and that we have a family of watermarks $\{\mathbf{w}_j\}$ that are fingerprints associated with the different users who purchase the rights to access $\mathbf{x}$. Before the watermarks are added to the host signal, every component of each $\mathbf{w}_j$ is scaled by an appropriate factor that corresponds to an amplification, i.e. $\mathbf{s}_j(k) = \alpha(k)\mathbf{w}_j(k)$, where we refer the the kth component of a vector $\mathbf{w}_j$ by $\mathbf{w}_j(k)$. One possibility for $\alpha(k)$ is to use the *just-noticeable-difference* (JND) based on human visual system models [13]. Corresponding to each user is a marked version of the content $\mathbf{t}_j = \mathbf{x} + \mathbf{s}_j$, which typically experiences additional distortion $\mathbf{z}_j$ that is due to such factors as compression and attacks made to remove the embedded fingerprints. We will denote the combination of the noise and the interference of the original signal by $\mathbf{d}_j = \mathbf{x} + \mathbf{z}_j$. We can thus assume that each user will be given a marked content $\mathbf{y}_j$, where

$$\mathbf{y}_j = \mathbf{x} + \mathbf{s}_j + \mathbf{z}_j = \mathbf{t}_j + \mathbf{z}_j = \mathbf{s}_j + \mathbf{d}_j. \tag{5.1}$$

Typically, the watermarks $\{\mathbf{w}_j\}$ are chosen to correspond to orthogonal noiselike signals [12], or are represented using a basis of orthogonal noiselike signals $\mathbf{u}_i$ via

$$\mathbf{w}_j = \sum_{i=1}^{B} b_{ij}\mathbf{u}_i, \tag{5.2}$$

where $b_{ij} \in \{0, 1\}$ or $b_{ij} \in \{\pm 1\}$ [51].

One important application of fingerprinting is identifying a user who is redistributing marked content $\mathbf{y}_j$ by detecting the watermark associated with the user to whom $\mathbf{y}_j$ was sold. By identifying a user, the content owner may be able to more closely monitor future actions of that user, or gather evidence supporting that user's illicit usage of the content. There are two different detection strategies that might arise in fingerprinting applications. They are differentiated by the presence or lack of the original content in the detection process. We will refer to *non-blind* detection as the process of detecting the embedded watermarks with the assistance of the original content $\mathbf{x}$, and refer to *blind* detection as the process of detecting the embedded watermarks without the knowledge of the original content $\mathbf{x}$. Non-blind fingerprint detection requires that the entity performing detection first identify the original version corresponding to the test image from a database of unmarked original images. This database can often be very large and requires considerable storage resources. Blind detection, on the other hand, offers more flexibility in detection, such as distributed detection scenarios. It does not require vast storage resources, and does not have the computational burden associated with image registration from a large database. In addition, restricting to non-blind detection may suffer the attacks of multiple ownership claims discussed in [88]. The problem can be easily overcome by blind detection [89]. However, unlike the non-blind detection scenario, in the blind detection scenario the host signal is unknown to the detector and often serves as a noise source that hinders the ability to detect the watermark. We note that there are other types of watermarking schemes that do not suffer from interference from unknown host signals [70, 90]. Their appropriateness for fingerprinting and anti-collusion capabilities are to be investigated and will be addressed in future work.

The detection of additive watermarks can be formulated as a hypothesis testing problem, where the embedded data is considered as the signal that is to be detected in the presence of noise. For the popular spread spectrum embedding [12, 13], the detection performance can be studied via the following simplified antipodal model:

$$\begin{cases} H_0 : y_i = -s_i + d_i \quad (i = 1, ..., N) & \text{if } b = -1 \\ H_1 : y_i = +s_i + d_i \quad (i = 1, ..., N) & \text{if } b = +1 \end{cases} \tag{5.3}$$

where $\{s_i\}$ is a deterministic spreading sequence (often called the *watermark*), b is the one bit to be embedded and is used to antipodally modulate s_i, d_i is the total noise, and N is the number of samples/coefficients to carry the hidden information. In non-blind detection, where the original source is available, d_i comes from the processing and/or attacks; in blind detection, d_i consists of the host media as well as distortion from processing and attacks. If d_i is modelled as i.i.d. Gaussian $\mathcal{N}(0, \sigma_d^2)$, the optimal detector is a (normalized) correlator [91] with a detection statistics T_N given by

$$T_N \quad = \quad \mathbf{y}^T \mathbf{s} / \sqrt{\sigma_d^2 \cdot \|\mathbf{s}\|^2} \tag{5.4}$$

where $\mathbf{y} = [y_1, ..., y_N]^T$, $\mathbf{s} = [s_1, ..., s_N]^T$ and $\|\mathbf{s}\|$ is the Euclidean norm of $\|\mathbf{s}\|$. Under the i.i.d. Gaussian assumption for d_i, T_N is Gaussian distributed with unit variance and a mean value

$$E(T_N) \quad = \quad b \cdot \sqrt{\|\mathbf{s}\|^2 / \sigma_d^2}. \tag{5.5}$$

If b is equally likely to be "-1" and "+1", the optimal (Bayesian) detection rule is to compare T_N with a threshold of zero to decide H_0 against H_1, in which case, the probability of error is $\mathcal{Q}(E(T_N))$, where $\mathcal{Q}(x)$ is the probability $P(X > x)$ of a Gaussian random variable $X \sim \mathcal{N}(0, 1)$. The error probability can be reduced by

raising the watermark-to-noise-ratio (WNR) $\|\mathbf{s}\|^2/(N\sigma_d^2)$, or increasing the length N of the spreading sequence. The maximum watermark power is generally determined by perceptual models so that the changes introduced by the watermark are below the *just-noticeable-difference* (JND) [13]. Assuming that both $\{s_i\}$ and $\{d_i\}$ are zero mean, σ_d^2 is estimated from the power of y_i and s_i, for example via $\hat{\sigma}_d^2 = (\|\mathbf{y}\|^2 - \|\mathbf{s}\|^2)/N$.

The i.i.d. Gaussian noise assumption is critical for the optimality of a correlator-type detector, but it may not reflect the statistical characteristics of the actual noise and interference. For example, the noise and interference in different frequency bands are different. In such a scenario, we should first normalize the observations $\{y_i\}$ by the corresponding noise standard deviation to make the noise distribution i.i.d. before taking the correlation [92]. That is,

$$T'_N = \sum_{i=1}^{N} \frac{y_i \cdot s_i}{\sigma_{d_i}^2} / \sqrt{\sum_{i=1}^{N} \frac{{s_i}^2}{\sigma_{d_i}^2}} \tag{5.6}$$

and

$$E(T'_N) = b\sqrt{\sum_{i=1}^{N} \frac{s_i^2}{\sigma_{d_i}^2}} \tag{5.7}$$

This can be understood as a weighted correlator with more weight given to less noisy components. Similarly, colored noise needs to be whitened before correlation. In general, an optimal detector can be derived using realistic distributions for noise and host interference [91]. In this chapter, we will use the correlator with normalized noise variance as described in (5.6).

Another model, used often for conveying ownership information [12, 13], leads to a similar hypothesis testing problem described by:

$$\begin{cases} H_0 : y_i = d_i & (i = 1, ..., N) \quad \text{if watermark is absent} \\ H_1 : y_i = s_i + d_i & (i = 1, ..., N) \quad \text{if watermark is present} \end{cases} \tag{5.8}$$

This is often referred as *On-Off Keying* (OOK). The detection statistics is the same as shown in (5.4) or (5.6). The threshold for distinguishing the two hypotheses is a classic detection problem, for which we can use a Bayesian rule or a Neyman-Pearson rule [91]. The probability of detection errors can be obtained accordingly.

5.2.2 Collusion Scenarios

When two parties with the same image (but fingerprinted differently) come together, they can compare the difference between the two fingerprinted images. Collusion attack generates a new image from the two fingerprinted images so that the traces of either fingerprint in the new image is removed or attenuated. For fingerprinting through additive embedding, this can be done by averaging the two fingerprinted images $\mathbf{y}_c = \lambda_1 \mathbf{y}_1 + \lambda_2 \mathbf{y}_2$ where $\lambda_1 + \lambda_2 = 1$, so that the energy of each of the fingerprints is reduced to ${\lambda_i}^2$ of the corresponding original. The requirement that $\lambda_1 + \lambda_2 = 1$ is necessary in order to retain the pixel magnitudes of the original image. As a result of this weighted average, the detection statistics with respect to the i-th fingerprint is scaled by a factor of λ_i. Alternatively, the new image can be formed by taking part of the pixels or transform coefficients from each of the two images

$$\mathbf{y}_c = \mathbf{\Lambda}\mathbf{y}_1 + (\mathbf{I} - \mathbf{\Lambda})\mathbf{y}_2$$

where $\mathbf{I}$ is the $N \times N$ identity matrix and $\mathbf{\Lambda} = diag(\lambda_1, \lambda_2, ..., \lambda_N)$ with $\lambda_i \in \{0, 1\}$. In terms of the effects on the energy reduction of the original fingerprints and the effect it has upon the detection performance, this alternating type of collusion is similar to the averaging type. For the simplicity of discussion, we will focus on the averaging type of collusion in this chapter.

Collusion can be extended to more than two parties. In a K-colluder averaging-

collusion the watermarked content signals $\mathbf{y}_j$ are combined according to $\sum_{j=1}^{K} \lambda_j \mathbf{y}_j$. The objective of each colluder is to avoid being detected, yet remain fair to his fellow colluders and retain good image quality. We will present two perspectives of fairness for which the colluders may try to combine their watermarks, and observe that, under realistic assumptions about the detection statistics for each user, both scenarios yield the optimal values for λ_j as $\lambda_j = 1/K$ for all j. In the theorem that follows, we consider that each user's watermarked content is undistorted by compressions and attacks.

Theorem 2. *Suppose that the content* $\mathbf{x}$ *is watermarked differently for each user to produce marked content* $\mathbf{y}_j = \mathbf{x} + \mathbf{s}_j$. *Assume that we have a group of* K *colluders who conspire by forming colluded version of the content*

$$\mathbf{y}_c = \sum_{j=1}^{K} \lambda_j \mathbf{y}_j, \tag{5.9}$$

where $\sum_j \lambda_j = 1$. *Further, if the probability of detecting the* j*th user's watermark,* $p_j(\lambda)$, *is such that all users have identical detection properties, i.e.* $p_j(\lambda) = p(\lambda)$ *for all* j, *and* $p(\lambda)$ *is a differentiable, strictly monotonic increasing function in* λ *whose derivative* p' *is also monotonic, then*

1. *Under the group fairness rule that the users seek to maximize the probability that all colluders are undetected, the optimal allocation of the scaling parameters is* $\lambda_j = 1/K$ *for all* j.

2. *Under the min-max fairness rule that the colluders will seek to minimize the maximum probability of a colluder being detected, the optimal allocation of the scaling parameters is* $\lambda_j = 1/K$ *for all* j.

Proof. (1): Obviously, the smaller in magnitude λ_{j_0} is, the less likely user j_0 will be detected. In order to incorporate fairness to the other users, we seek to minimize

the probability that none of the colluders are detected. Suppose that the probability of detecting $\mathbf{w}_{j_0}$ given that user j_0 scales his watermark by λ_{j_0} is denoted by $p_{j_0}(\lambda_{j_0})$. It is reasonable to assume that each watermark has the same detection properties as the others for the same scaling factor, i.e. $p_j(\lambda) = p(\lambda)$ for all j and for all λ. It is also reasonable to assume that $p(\lambda)$ and its derivative p' is monotonic since the effect of scaling $\mathbf{y}$ by λ on a correlation detection statistics is to scale the correlation statistics by λ. The probability that all colluders are undetected is

$$G(\lambda) = \prod_{j=1}^{K} (1 - p(\lambda_j)) = \prod_{j=1}^{K} g(\lambda_j).$$

We would like to maximize $G(\lambda)$ subject to the constraint $\sum_j \lambda_j = 1$. To simplify the derivation, define $h(\lambda) = \log g(\lambda)$. Maximizing $G(\lambda)$ is equivalent to maximizing $H(\lambda) = \sum_{j=1}^{K} h(\lambda_j)$. Now define the LaGrangian Dual function $L(\lambda, \mu)$ by

$$L(\lambda, \mu) = H(\lambda) + \mu \left(\sum_j \lambda_j - 1 \right).$$

Taking the derivative of $L(\lambda, \mu)$ with respect to λ_j gives

$$\frac{dL}{d\lambda_j} = h'(\lambda_j) + \mu$$

Setting this to 0, and observing that the conditions on p and p' are sufficient for $(h')^{-1}$ to exist, we may conclude that $\lambda_j = -(h')^{-1}(\mu)$ for all j and hence $\lambda_j = 1/K$ for all j.

(2): We want to minimize the maximum probability of a colluder getting detected:

$$\hat{p} = \min_{\lambda} \max_{j} p(\lambda_j).$$

Observe that when we fix λ, the maximum $p(\lambda_j)$ corresponds to the maximum λ_j, which we denote λ_{max}. Since $\lambda_{max} \geq 1/K$, we have that $\hat{p} \geq p(1/K)$. Observe that

the choice of $\lambda_j = 1/K$ for all j achieves the lower bound and hence the min-max assignment is the choice of λ that achieves this lower bound. □

5.3 Orthogonal Modulation and Anti-Collusion

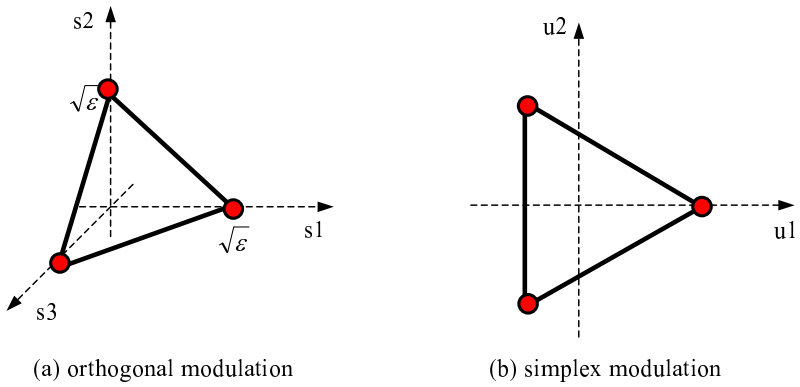
In this section we will focus on the methods of orthogonal, and simplex modulation [93] for embedding unique fingerprints to multiple copies of images. We also observe that biorthogonal modulation is not appropriate for fingerprinting.

In orthogonal modulation, there are v orthogonal signals $\mathbf{s}_j$ that are used to convey $B = \log_2 v$ bits by inserting one of the v signals into the host signal. These B bits can be used to identify the n users by identifying a B-bit ID sequence with each user, and therefore we have $n = v$. The detector determines the B information bits by performing the correlation of the test signal with each of the v signals, and decides the signal that has the largest correlation above a minimum threshold. The corresponding B-bit index of this signal is the ID sequence of the user. Typically, v correlations are used to determine the embedded signal, and the computational complexity associated with performing v correlations is considered one of the drawbacks of orthogonal modulation. In Section 5.3.2, we present an improved detection strategy that cuts the computational complexity from $\mathcal{O}(v)$ to $\mathcal{O}(\log v)$.

An additional drawback for using orthogonal modulation in data embedding is the large number of orthogonal signals needed to convey B bits. In many situations, it might not be possible to find 2^B orthogonal signals in the content. In audio applications, it might be desirable to periodically repeat a watermark embedding in the content in order to fingerprint clips from the audio. In this case the number of orthogonal basis signals available is limited by the sample rate. For example, if we

repeat a watermark every second in audio with a 44.1kHz sample rate, then we can allow for at most $44,100$ users to purchase the content if orthogonal modulation is used in fingerprinting. Although other media, such as images and video, might have more points per embedding period, many of these degrees of freedom will be lost since embedding should only take place in perceptually significant components [12]. In particular, some content, such as smoothly textured images and binary images, are known to have a significantly lower embedding rate than what is suggested by the amount of points in the image. Further, the necessary bookkeeping and storage of the $v = 2^B$ basis vectors is another drawback of orthogonal modulation. In Section 5.4, we build watermarks using code modulation that are able to handle more users than orthogonal modulation for the same amount of orthogonal vectors.

A variation of orthogonal modulation is biorthogonal modulation, which consists of v orthogonal basis signals and their negatives [93]. Biorthogonal modulation has the advantage that it is able to convey $B + 1$ bits using v orthogonal basis. A second variation of orthogonal modulation is simplex modulation. Simplex modulation makes more efficient usage of the energy by maximizing the amount of angular separation for v signals in $v - 1$ dimensional space [93]. The constellation points for v-ary simplex modulation are all spaced at equal distances on the $v - 1$ dimensional sphere. The simplex signals have the property that all pairs of signals have the same correlation. The $v - 1$ dimensional constellation points for v-ary simplex modulation can be calculated by subtracting the mean from the constellation points for v-ary orthogonal modulation. With the same amount of energy, v-ary simplex modulation increases the separation between constellation points when compared with v-ary orthogonal modulation. For reference, in Figure 5.1, we depict an example of the constellation points for 3 signals for the orthogonal



(a) orthogonal modulation (b) simplex modulation

Figure 5.1: An example the constellation points for $v = 3$ orthogonal and simplex modulation.

and simplex modulation schemes.

5.3.1 Anti-Collusion Performance

We now study the behavior of the different modulation schemes under averaging collusion attacks. We start by discussing the effect of collusion upon biorthogonal modulation. In biorthogonal modulation, it is possible that two users with the watermarks $\mathbf{s}_j$ and $-\mathbf{s}_j$ might decide to collude. The average of these two watermarks would completely erase the presence of any identifying watermark. Therefore, it is not appropriate for use in fingerprinting applications, and we shall not consider it in the remainder of the chapter.

We can study the effect of collusion on orthogonal and simplex modulation by calculating the distance between the constellation points and averages of the constellation points. Additionally, since the goal of collusion is to create a new content whose watermarks have been sufficiently attenuated that they are undetectable, we would like to calculate the distance between the averages of the constellation

points and the origin.

We start by considering the case of orthogonal modulation. Suppose each watermark is embedded using $\mathcal{E}$ energy. If we average K watermarks, then the distance from the colluded mark to any of the watermarks used in forming it is $\sqrt{\mathcal{E}(K-1)/K}$. The distance from the colluded mark to any of the other watermarks not used in the collusion is $\sqrt{\mathcal{E}(K+1)/K}$. Further, the distance of the colluded mark from the origin is $\sqrt{\mathcal{E}/K}$. Thus, as K increases, the identifying watermarks in the colluded mark will become harder to detect.

Next, we calculate the distances for v-ary simplex modulation. In order to calculate the various distances, we will describe the watermark signals $\mathbf{w}_j$ in terms of the standard v-dimensional basis, in which case

$$\begin{aligned}
\mathbf{w}_1 &= \left(\frac{v-1}{v}, \frac{-1}{v}, \frac{-1}{v}, \cdots, \frac{-1}{v}\right) \\
\mathbf{w}_2 &= \left(\frac{-1}{v}, \frac{v-1}{v}, \frac{-1}{v}, \cdots, \frac{-1}{v}\right) \\
&\vdots \\
\mathbf{w}_v &= \left(\frac{-1}{v}, \frac{-1}{v}, \frac{-1}{v}, \cdots, \frac{v-1}{v}\right).
\end{aligned}$$

Each $\mathbf{w}_j$ has amplitude $\beta = \sqrt{(v-1)/v}$, and we may now normalize the $\mathbf{w}$'s to produce $\mathbf{s}_j = \sqrt{\mathcal{E}}\mathbf{w}_j/\beta$. We define the colluded mark $\mathbf{s}_c$ by

$$\mathbf{s}_c = \frac{1}{K}\sum_{j=1}^{K} \mathbf{s}_j.$$

The distance between any two $\mathbf{w}$ vectors is easily seen to be $\sqrt{2}$. By scaling by $\sqrt{\mathcal{E}}/\beta$ we get that any two simplex constellation points are separated by a distance of $\sqrt{2\mathcal{E}v/(v-1)}$. The distance of $\mathbf{s}_c$ from the origin is

$$\|\mathbf{s}_c\| = \frac{\sqrt{\mathcal{E}}}{\beta}\sqrt{\frac{1}{K} - \frac{1}{v}}. \tag{5.10}$$

5.3.2 Efficient Detection Strategy for Orthogonal Modulated Fingerprints

The classical method for estimating which signal was embedded in the host signal is done via v correlators, and determines the B bit message that identifies which user's watermark was present. This has been considered a major drawback of the method of orthogonal modulation [12, 94]. In this section we present an algorithm that dramatically reduces the computation needed to detect which watermarks are present in a host signal. The algorithm that we present is an example of a tree-based search algorithm used in group testing [95–98].

Suppose that K colluders are involved in forming a colluded signal $\mathbf{y}_c$. We desire to identify the basis vectors of these K colluders. If there are a total of $n = v$ orthogonal signals, one for each user, then testing for these K colluders' basis vectors can be done by correlating the received watermark with each of the n orthogonal signals. This can be prohibitive for large n as it leads to significant computational requirements.

For a set $A = \{\mathbf{w}_j\}_{j \in J}$ where J is an indexing set, we define the sum of A by $SUM(A) = \sum_{j \in J} \mathbf{w}_j$. We start by considering the case of detecting 1 watermark. Let us denote by $S = \{\mathbf{w}_j\}$ the set of orthogonal watermark signals, and suppose the test signal is $\mathbf{y}$. Suppose that we break S into two complementary subsets S_0 and S_1. If we correlate the test signal $\mathbf{y}$ with $SUM(S_0)$ then the correlation will satisfy

$$\langle \mathbf{y}, \sum_{\mathbf{w}_j \in S_0} \mathbf{w}_j \rangle = \sum_{j \in J} \langle \mathbf{y}, \mathbf{w}_j \rangle, \tag{5.11}$$

where $\langle y, w \rangle$ denotes a correlation process, such as is described in (5.4). If the one watermark we desire to detect belongs to the set S_0 then $\langle \mathbf{y}, SUM(S_0) \rangle$ will experience a large contribution from that one waveform, and all the other terms

will have small values. If this watermark is not present in S_0, then $\langle \mathbf{y}, SUM(S_0) \rangle$ will consist only of small contributions. Therefore, if we test two sets S_0 and S_1 such that $S_1 = S \backslash S_0$, then we are guaranteed to get a large value in one of the two correlations with the sum of the basis vectors.

We can repeat this idea by further decomposing S_0 and S_1 if they pass a threshold test. This idea can be extended to detecting the presence of K orthogonal signals. At each stage we test two sets S_0 and S_1, and if a set passes a threshold test, then we further decompose it. We use this idea to develop a recursive detection algorithm for detecting the presence of K orthogonal signals in a test signal $\mathbf{y}$. There are many possible choices for dividing S into S_0 and S_1 in such an algorithm. In Algorithm 6 we have chosen S_0 such that $|S_0| = 2^{\lceil \log_2 |S| \rceil - 1}$, which is the largest power of 2 less than $|S|$. We chose $|S_0| = 2^{\lceil \log_2 |S| \rceil - 1}$ in order to ensure that at least one set had an amount of elements that was a power of 2, which facilitated an easier calculation of the computational bound in Lemma 10. Another possible choice, which might be more efficient, would be to take S_0 such that $|S_0| = \lceil |S|/2 \rceil$.

We now make some observations about the computational performance of this algorithm. First, we address the number of correlations that must be performed to identify K signals in a test signal $\mathbf{y}$. Let us denote by $C(n, K)$ the number of correlations needed in Algorithm 6 to identify K signals from a set S of n orthogonal signals. Lemma 10 provides a bound $C(n, K)$.

Lemma 10. *The number of correlations $C(n, K)$ needed in Algorithm 6 satisfies*

$$C(n, K) \leq 2 \left(-1 + K \left(\log_2 (2^{\lceil \log_2 n \rceil} / K) + 1 \right) \right). \tag{5.12}$$

Proof. The proof follows a standard counting argument for tree-based algorithms [28], and similar proofs can be found in [96–99]. We break the proof into two parts.

Algorithm: $EffDet(\mathbf{y}, S)$

Divide S into two sets S_0 and S_1, where $|S_0| = 2^{\lceil \log_2 |S| \rceil - 1}$, and $S_1 = S \backslash S_0$;

Calculate $\mathbf{e}_0 = SUM(S_0)$ and $\mathbf{e}_1 = SUM(S_1)$;

Calculate $\rho_0 = \langle \mathbf{y}, \mathbf{e}_0 \rangle$ and $\rho_1 = \langle \mathbf{y}, \mathbf{e}_1 \rangle$;

if $\rho_0 > \tau$ **then**

  if $|S_0| = 1$ **then**

    output S_0 ;

  else

    $EffDet(\mathbf{y}, S_0)$;

  end

end

if $\rho_1 > \tau$ **then**

  if $|S_1| = 1$ **then**

    output S_1 ;

  else

    $EffDet(\mathbf{y}, S_1)$;

  end

end

Algorithm 6: Efficient detection algorithm, $EffDet(\mathbf{y}, S)$

First, we will calculate a bound for $n = 2^r$, and then we shall extend to general n. We begin by observing that corresponding to the algorithm is a tree, and can associate with each internal node either a S_0 or a S_1 set, as defined in Algorithm 6. We may also associate with each internal node a flag that identifies whether the magnitude of the correlation of the test signal with the sum of the basis vectors in the node's set is larger than a threshold. We shall call such internal nodes positive-nodes, and we assume the root node is always positive. Examples of the decision tree associated with the algorithm are presented in Figure 5.2.

If $n = 2^r$, the number of levels in the tree is $r + 1$, where we count the root node as the first level, and the leaf nodes as belonging to the $r + 1$ level. Define $L = \lceil \log_2 K \rceil$. We now perform a standard counting of the worst-case amount of positive nodes. The first level that is able to have K positive-nodes is level $L + 1$, while the number of positive-nodes in level $j < L+1$ is at most 2^{j-1}. There are two cases to consider when bounding the number of positive nodes. First, we look at the positive nodes from level 1 to level L. This is at most $1+2+\cdots+2^{L-1} = 2^L-1$. Next, from level $L+1$ to level r we have at most $(r-L)K$ positive nodes and thus the total number of positive nodes $T(n, K)$ from root to level r is bounded by

$$\begin{aligned} T(n,K) &\leq 2^L - 1 + (r-L)K \\ &= K2^{L-\log_2 K} - 1 + K\left(\log_2(n/K) - L + \log_2 K\right) \\ &\leq -1 + K\left(\log_2(n/K) + 1\right), \end{aligned}$$

where we have used the observation that $L - \log_2 K \in [0, 1)$ and hence $2^{L-\log_2 K} - L + \log_2 K \leq 1$. Since each positive nodes corresponds to two correlations, the total amount of correlations is bounded by

$$C(n,K) \leq 2\left(-1 + K\left(\log_2(n/K) + 1\right)\right)$$

for $n = 2^r$. The general case is addressed by observing that $C(n+1, K) \geq C(n, K)$. This implies $C(2^{\lceil \log_2 n \rceil}, K) \geq C(n, K)$ which, when combined with the previous bound for $C(n, K)$, gives the general case

$$C(n,K) \leq 2\left(-1 + K\left(\log_2(2^{\lceil \log_2 n \rceil}/K) + 1\right)\right).$$

□

In particular, the result of this lemma gives us that if we were trying to detect a single signal, then need to perform at most $2(\lceil \log_2 |S| \rceil - 1)$ correlations as

opposed to $|S|$ in a traditional implementation. Also, as K becomes larger, the improvement in the amount of correlations performed decreases since it becomes necessary to perform correlations for multiple branches of the tree. For example, if $K = n$, then simply correlating with each of the n orthogonal signals is more efficient than Algorithm 6.

We mentioned earlier that Algorithm 6 is an example of a group testing algorithm. Algorithm 6 does not require knowledge of the the amount of colluders, or *defective items* in the parlance of group testing. When the amount of defective items is known a priori, it is possible to develop more efficient testing procedures [98]. In fact, in [96] a competitive group testing algorithm is designed that attempts to estimate the number of defective items K, and yields a 1.65-competitive algorithm that is asymptotically more efficient than the simple bisecting algorithm presented in Algorithm 6 or in [96].

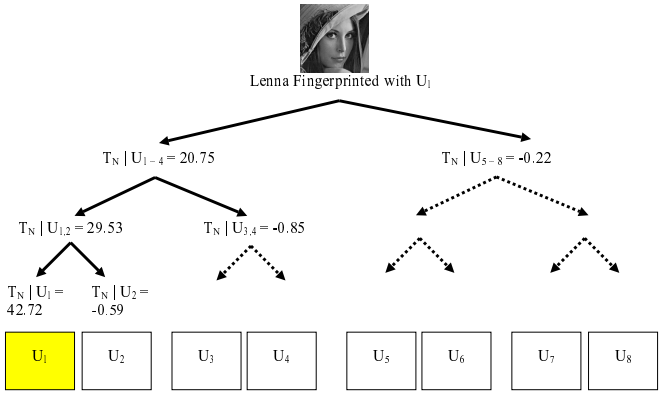
We also observe that although we have achieved an improvement in computational efficiency, this comes at a tradeoff in detector variance. When we calculate the correlation with the sums of basis vectors, we get many small, noisy contributions from correlating the test signal with signals not present in the test signal, as in (5.11). One approach to reduce this increased noise variance, is to modify the algorithm by dividing S into more, smaller subsets. Another approach to handle the errors introduced due to the increased variance would be to use searching algorithms robust to errors, such as described in [99].

5.3.3 Experiments on Efficient Detection of Orthogonal Modulated Fingerprints

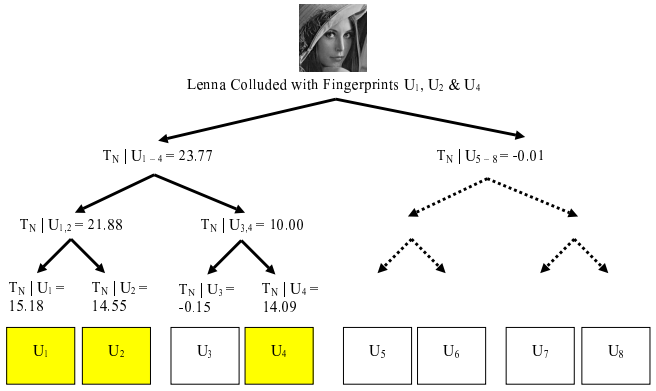
We desired to study the performance of the efficient detection algorithm, and the effect that collusion had on the detection statistics. In our experiments, we used an additive spread spectrum watermarking scheme similar to that in [13], where a perceptually weighted watermark was added to DCT coefficients with a block size of 8×8. The detection of the watermark is performed without the knowledge of the host image via the detection statistics as shown in (5.6). The 512×512 Lenna is used as the host image for fingerprinting, and the fingerprinted images had no visible artifacts with an average PSNR of 41.2dB. Figure 5.2 illustrates the process of identifying colluders out of 8 users using the efficient detection algorithm (Algorithm 6). The detection statistics are averaged over 10 different sets of watermarks, and each set has 8 mutually uncorrelated spread spectrum watermarks for 8 users. These watermarks are generated via a psedudo-random number generator and used as an approximate orthogonal basis in orthogonal modulation.

Figure 5.2(a) shows the process of detecting colluders from an image with user 1's fingerprint embedded. The notation "$T_N \| U_?$" denotes the detection statistics when correlating the test image with the sum of the fingerprints $U_?$. Detection statistics close to zero indicate the unlikely contributions from the corresponding fingerprints, and the branches of the detection tree below them, indicated by dotted lines, are not explored further. The number of correlations performed is 6. Figure 5.2(b) shows the process of detecting colluders from an image colluded from user 1,user 2, and user 4's fingerprinted images. The number of correlations performed is 8.

We see from Figure 5.2(a) that the detection statistics when correlating with a



(a)



(b)

Figure 5.2: Detection trees for identifying colluders using Algorithm 1. The images for different users are fingerprinted via orthogonal modulation. The fingerprints of colluders are indicated by shadowed boxes U_j. The notation "$T_N \| U_?$" denotes the detection statistics from correlating the test image with the sum of the fingerprints $U_?$. Detection statistics close to zero indicate the unlikely contributions from the corresponding fingerprints, and the branches of the detection tree below them, indicated by dotted lines, will not be explored.

sum of a larger number of basis vectors is smaller than that with a smaller amount of basis vectors. This reflects the noisy contributions from the basis vectors that are present in the sum of basis vectors but are not present in the test image. We mentioned this phenomena earlier in Section 5.3.2. Since the detection statistics we use has its variance normalized to 1, the noisy contributions lower the detection statistics values. We also observe in Figure 5.2(b) a decrease in the detection statistics in images colluded by more users.

5.4 Code Modulation Embedding and Anti-Collusion Codes

In the previous section, we mentioned that a drawback of the usage of orthogonal signaling is the large amount of basis vectors needed to convey user information. In this section we will present another form of modulation that may be used to convey more bits of information for a given amount of basis vectors than orthogonal modulation. Therefore, we are able to accommodate more users for the same amount of orthogonal signals than orthogonal modulation. We will use this modulation technique, in conjunction with appropriately designed codewords, known as anti-collusion codes, to construct a family of watermarks that have the ability to identify members of the colluding set of users.

In code modulation, there are v orthogonal basis signals $\{\mathbf{u}_j\}$, and information is encoded into a watermark signal $\mathbf{w}_j$ via

$$\mathbf{w}_j = \sum_{j=1}^{v} b_{ij}\mathbf{u}_i, \tag{5.13}$$

where $b_{ij} \in \{0, 1\}$ or $b_{ij} \in \{\pm 1\}$. The first of the two possibilities for choosing the values of b_{ij} corresponds to on-off keying (OOK) while the second choice of $\{\pm 1\}$

corresponds to an antipodal form [93]. If $b_{ij} = 0$, this is equivalent to having no contribution in the $\mathbf{u}_i$ direction. At the detector side, the determination of each b_{ij} is done by correlating with the $\mathbf{u}_i$, and comparing against a decision threshold. In either case, $B = v$ bits of information are conveyed using v orthogonal basis vectors. If $\mathcal{E}$ energy is allocated to each watermark, then the OOK form of binary code modulation will devote more energy to fewer basis functions than the antipodal form, which evenly divides $\mathcal{E}$ energy amongst all of the v different orthogonal basis functions.

We assign a different bit sequence $\{b_{ij}\}$ for each user u_k. We may view the assignment of the bits b_{ij} for different watermarks in a matrix $\mathbf{B}$, which we call the *derived* code matrix, where different columns of $\mathbf{B}$ contain *derived* codevectors for different users. This viewpoint allows us to capture the orthogonal, simplex, and coded modulation cases for watermarking. For example, the identity matrix describes the orthogonal signaling case since the jth user is only associated with one signal vector $\mathbf{u}_j$. In the following section, we shall design a code matrix $\mathbf{C}$ whose elements are either 0 or 1. By applying a suitable mapping that depends on whether the OOK or antipodal form of code modulation is used, the code matrix $\mathbf{C}$ is used to derive the matrix $\mathbf{B}$ that is used in forming the watermark signals.

In binary code modulation, if we average two watermarks, $\mathbf{w}_1$ and $\mathbf{w}_2$ corresponding to bit sequences b_{j1} and b_{j2}, then when $b_{j1} \neq b_{j2}$ the contributions attenuate or cancel depending on whether the OOK or antipodal form is used. However, when $b_{j1} = b_{j2}$ the contributions do not attenuate. For example, if antipodal code modulation is used, the result of averaging two watermark signals is that many of the components will still have $\sqrt{\mathcal{E}/v}$ amplitude, which is identical to the amplitude prior to collusion, while other components will have 0 amplitude.

When we average K watermarks, those components in the bit sequences that are all the same will not experience any cancellation, and their amplitude will remain $\sqrt{\mathcal{E}/v}$, while others will experience diminishing (though not necessarily complete cancellation).

5.4.1 Anti-Collusion Codes

In this section we design a family of codevectors $\{\mathbf{c}_j\}$ whose overlap with each other can identify groups of colluding users. A similar idea was proposed in [100], where projective geometry was used to construct such code sequences. As we will explain in this section, our proposed code construction makes more efficient usage of the basis vectors than the codes described in [100].

For this section, we describe codes using the binary symbols $\{0, 1\}$. These codevectors are mapped to *derived* codevectors by a suitable mapping depending on whether the OOK or antipodal form of binary code modulation is used for watermarking. For example, when used in the antipodal form, the binary symbols $\{0, 1\}$ are mapped to $\{-1, 1\}$ via $f(x) = 2x - 1$.

We assume, when a sequence of watermarks is averaged and detection is performed, that the detected binary sequence is the logical AND of the codevectors $\mathbf{c}_j$ used in constructing the watermarks. For example, when the watermarks corresponding to the codevectors (1110) and (1101) are averaged, the output of the detector is (1100). When we perform 2 or more averages, this assumption might not necessarily hold since the average of many 1's and a few 0's may produce a decision statistic large enough to pass through the detector as a 1. We discuss the behavior of the detector in these situations further in Section 5.4.2, and detail approaches to improve the validity of the AND assumption.

We want to design codes such that when K or fewer users collude, we can identify the colluders. We prefer shorter codes since for embedded fingerprints longer codes would distribute the fingerprint energy over more basis vectors, which would lead to a higher error rate in the detection process. In order to identify colluders, we first require that there is some non-zero component remaining in the code when the codes for these K colluders are combined. Secondly, we require that there are no repetitions in the different combinations of K or fewer codevectors. We will call codes that satisfy these properties anti-collusion codes. In the definition that follows, we provide a generic definition in terms of semigroups [101], and then specify the relevant case that we use in this chapter.

Definition 3. *Let G be a semigroup with a binary operation $\star$. A code $\mathcal{C} = \{\mathbf{c}_1, \cdots, \mathbf{c}_n\}$ of vectors belonging to G^v is called a K-resilient $(G, \star)$ anti-collusion code, or a $(G, \star)$ ACC when any subset of K or fewer codevectors combined element-wise under $\star$ is non-zero and distinct from the element-wise $\star$ of any other subset of K or fewer codevectors. When $G = \{0, 1\}$ and $\star$ is the logical AND, a K-resilient $(G, \star)$ ACC is simply called a K-resilient AND-ACC.*

We first present a $(n-1)$-resilient AND-ACC. Let $\mathcal{C}$ consist of all n-bit binary vectors that have only a single 0 bit. For example, when $n = 4$, $\mathcal{C} = \{1110, 1101, 1011, 0111\}$. It is easy to see when $K \leq n-1$ of these vectors are combined under AND, that this combination is unique. This code has cardinality n, and hence can produce at most n differently watermarked media. We refer to this code as the *trivial* AND-ACC for n users.

It is desirable to shorten the codelength to squeeze more users into fewer bits since this would require the use and maintenance of fewer orthogonal basis vectors. To do this, we need to give up some resiliency. We now present a construction

of a K-resilient AND-ACC that requires $\mathcal{O}(\sqrt{n})$ basis vectors for n users. This construction uses balanced incomplete block designs [102]:

Definition 4. *A (v, k, λ) balanced incomplete block design (BIBD) is a pair $(\mathcal{X}, \mathcal{A})$, where $\mathcal{A}$ is a collection of k-element subsets (blocks) of a v-element set $\mathcal{X}$, such that each pair of elements of $\mathcal{X}$ occur together in exactly λ blocks.*

A (v, k, λ)-BIBD has $n = \lambda(v^2 - v)/(k^2 - k)$ blocks. Corresponding to a block design is the $v \times n$ incidence matrix $\mathbf{M} = (m_{ij})$ defined by

$$m_{ij} = \begin{cases} 1 & \text{if the } i\text{th element belongs to the } j\text{th block,} \\ 0 & \text{otherwise.} \end{cases}$$

If we define the codematrix $\mathbf{C}$ as the bit-complement of $\mathbf{M}$, and assign the codevectors $\mathbf{c}_j$ as the columns of $\mathbf{C}$, then we have a $(k-1)$-resilient AND-ACC. Our codevectors are therefore v-dimensional, and we are able to accommodate $n = \lambda(v^2 - v)/(k^2 - k)$ users with these v basis vectors. Assuming that a BIBD exists, for n users we therefore need $v \approx \mathcal{O}(\sqrt{n})$ basis vectors.

Theorem 3. *Let $(\mathcal{X}, \mathcal{A})$ be a $(v, k, 1)$-BIBD, and $\mathbf{M}$ the corresponding incidence matrix. If the codevectors are assigned as the bit complement of the columns of $\mathbf{M}$, then the resulting scheme is a $(k-1)$-resilient AND-ACC.*

Proof. We prove the theorem by working with the blocks A_j of the BIBD. The bitwise complementation of the column vectors corresponds to complementation of the sets $\{A_j\}$. We would like for $\cap_{j \in J} A_j^C$ to be distinct over all sets J with cardinality less than or equal to $k-1$. By De Morgan's Law, this corresponds to uniqueness of $\cup_{j \in J} A_j$ for all sets J with cardinality less than or equal to $k-1$. Suppose we have a set of $k-1$ blocks $A_1, A_2, \cdots, A_{k-1}$, we must show that there does not exist another set of blocks whose union produces the same set. There are

two cases to consider. First, assume there is another set of blocks $\{A_i\}_{i\in I}$ with $\cup_{j\in J}A_j = \cup_{i\in I}A_i$ such that $I\cap J = \emptyset$ and $|I| \leq k-1$. Suppose we take a block A_{i_0} for $i_0 \in I$. Then A_{i_0} must share at most one element with each A_j, otherwise it would violate the $\lambda = 1$ assumption of the BIBD. Therefore, the cardinality of A_i is at most $k-1$, which contradicts the requirement that each block have k elements. Thus, there does not exist another set of blocks $\{A_i\}_{i\in I}$ with $\cup_{j\in J}A_j = \cup_{i\in I}A_i$ and $I\cap J = \emptyset$. Next, consider $I\cap J \neq \emptyset$. If we choose $i_0 \in I\backslash(I\cap J)$ and look at A_{i_0}, then again we have that A_{i_0} can share at most 1 element with each A_j for $j \in J$, and thus A_{i_0} would have fewer than k elements, contradicting the fact that A_{i_0} belongs to a $(v,k,1)$-BIBD. Thus, $\cup_{j\in J}A_j$ is unique. □

We now present an example that is built from a $(7,3,1)$-BIBD. The $(7,3,1)$-BIBD is an example of projective geometry and is often called the seven-point plane [103]. There are many different possible seven-point planes that can be constructed by permuting the labels of the points. If we take the bit-complement of the incidence matrix corresponding to one of these possible $(7,3,1)$-BIBDs we get:

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}. \tag{5.14}$$

This code requires 7 bits for 7 users and provides 2-resiliency since any two column vectors share a unique pair of 1 bits. Each column vector $\mathbf{c}$ of $\mathbf{C}$ is mapped to $\{\pm 1\}$

by $f(x) = 2x-1$. The code modulated watermark is then $\mathbf{w} = \sum_{i=1}^{v} f(c_i)\mathbf{u}_i$. When two watermarks are averaged, the locations where the corresponding AND-ACC agree and have a value of 1 identify the colluding users. For example, let

$$\mathbf{w}_1 = -\mathbf{u}_1 - \mathbf{u}_2 + \mathbf{u}_3 - \mathbf{u}_4 + \mathbf{u}_5 + \mathbf{u}_6 + \mathbf{u}_7 \quad (5.15)$$

$$\mathbf{w}_2 = -\mathbf{u}_1 + \mathbf{u}_2 - \mathbf{u}_3 + \mathbf{u}_4 + \mathbf{u}_5 - \mathbf{u}_6 + \mathbf{u}_7 \quad (5.16)$$

be the watermarks for the first two columns of the above $(7, 3, 1)$ code, then $(\mathbf{w}_1 + \mathbf{w}_2)/2$ has coefficient vector $(-1, 0, 0, 0, 1, 0, 1)$. The fact that a 1 occurs in the 5th and 7th location uniquely identifies user 1 and user 2 as the colluders.

The $(7, 3, 1)$ example that we presented had no improvement in bit efficiency over the trivial AND-ACC for 7 users, and it had less collusion resilience. A useful metric for evaluating the efficiency of an AND-ACC for a given resiliency is its rate $R = v/n$, which describes the amount of basis vectors needed per user. AND-ACCs with lower rates are better. For codes (v, k, λ)-BIBD AND-ACC, their rate is $R = (k^2-k)/(\lambda(v-1))$. Therefore, the efficiency of an AND-ACC built from BIBDs improves as the codelength v becomes larger. By Fisher's Inequality [102], we also know that $n \geq v$ for a (v, k, λ)-BIBD, and thus $R \leq 1$ using the BIBD construction. In contrast, the k-resilient construction in [100] has rate much larger than 1, and thus requires more spreading sequences (or marking locations) to accommodate the same amount of users as our scheme. It is possible to use the collusion-secure code constructions of [11] in conjunction with code modulation for embedding. However, the construction described in [11] has codelength $\mathcal{O}(\log^4 n \log^2(1/\epsilon))$, where $\epsilon < 1/n$ is the decision error probability. This codelength is considerably large for small error probabilities and practical n values. For example, when $n = 2^{10}$, the codelength of [11] is on the order of 10^4 or higher, while the codelength for our proposed AND-ACC is on the order of 10^2. Additionally, for the same amount

of users, the use of code modulation watermarking with an AND-ACC constructed using a $(v, k, 1)$-BIBD requires less spreading sequences than orthogonal signaling. A code modulation scheme would need v orthogonal sequences for $n = (v^2 - v)/(k^2 - k)$ users, while orthogonal signaling would require n sequences.

In general, (v, k, λ)-BIBDs do not necessarily exist for an arbitrary choice of v and k. The condition that n must be an integer restricts some possibilities for v and k, and for a given (v, k, λ) triple there may not exist a (v, k, λ)-BIBD. We may, however, construct infinite families of BIBDs. For example, $(v, 3, 1)$ systems (also known as Steiner triple systems) are known to exist if and only if $v \equiv 1$ or $3 \pmod 6$. The Bose construction builds Steiner triple systems when $v \equiv 3 \pmod 6$, and the Skolem construction builds Steiner triple systems when $v \equiv 1 \pmod 6$ [104]. Another approach to constructing BIBDs is to use d-dimensional projective and affine geometry over Z_p, where p is of prime power. Projective and affine geometries yield $((p^{d+1} - 1)/(p - 1), p + 1, 1)$ and $(p^d, p, 1)$ BIBDs [102, 105]. Techniques for constructing these and other BIBDs can be found in [106]. Finally, we mention that other combinatorial objects, such as packing designs and pairwise balanced designs, have very similar properties to BIBD, and may be used to construct AND-ACC where the codevectors do not all have the same weight. The construction and use of AND-ACC built from other combinatorial objects is beyond the scope of this chapter.

Given that the output of the detector is a vector $\mathbf{\Gamma} = (\Gamma_1, \Gamma_2, \cdots, \Gamma_n)$, we must determine who the colluders are. In practice, the noise in the detection process causes $\mathbf{\Gamma}$ to have errors, and therefore we may not be able to exactly identify the colluders. Instead, what we would like to do is use $\mathbf{\Gamma}$ to determine a *suspicious* set from the entire user set. In Algorithm 7, we determine a vector

Algorithm: $SuspectAlg(\mathbf{\Gamma})$

$\mathbf{\Phi} = \mathbf{1}$;

Define J to be the set of indices where $\Gamma_j = 1$;

for $t = 1$ ***to*** $|J|$ **do**

  $j = J(t)$;

  Define $\mathbf{e}_j$ to be the jth row of $\mathbf{C}$;

  $\mathbf{\Phi} = \mathbf{\Phi} \cdot \mathbf{e}_j$;

end

Algorithm 7: Algorithm $SuspectAlg(\mathbf{\Gamma})$, which determines the vector $\mathbf{\Phi}$ that describes the suspect set.

$\mathbf{\Phi} = (\Phi_1, \Phi_2, \cdots, \Phi_n) \in \{0,1\}^n$ that describes the suspicious set via the location of components whose value are 1. Thus, if $\Phi_j = 1$, then the jth user is suspected of colluding. In the algorithm, we denote the jth row vector of $\mathbf{C}$ by $\mathbf{e}_j$, and use the fact that the element-wise multiplication "$\cdot$" of the binary vectors corresponds to the logical AND operation. The algorithm starts with $\mathbf{\Gamma}$ and $\mathbf{\Phi} = \mathbf{1}$, where $\mathbf{1}$ is the n dimensional vector consisting of all ones. The algorithm then uses the indices where $\mathbf{\Gamma}$ is equal to 1, and updates $\mathbf{\Phi}$ by performing the AND of $\mathbf{\Phi}$ with the rows of the code matrix $\mathbf{C}$ corresponding to indices where $\mathbf{\Gamma}$ is 1. The algorithm starts with the entire group as the suspicious set, and uses the components of Γ that are equal to 1 to further narrow the suspicious set.

5.4.2 Detector Design and Performance

In this section we focus on the detector involved in collusion for binary code modulation. In order for the detector to determine whether $\mathbf{u}_i$, $-\mathbf{u}_i$, or neither exists in the test signal $\mathbf{y}$, we assume that the detector will correlate $\mathbf{y}$ with $\mathbf{u}_i$. It suffices to

consider the detector for a single basis vector, and therefore we drop the subscripts and just use $\mathbf{u}$.

Suppose that a codevector $\mathbf{c}_j$ has weight $\omega = wt(\mathbf{c}_j)$. In the OOK case the remaining $v-\omega$ positions would be zeros, while in the antipodal case the remaining $v-\omega$ positions would be -1. If we allocate $\mathcal{E}$ energy to this codevector, then the OOK case would use $\mathcal{E}/\omega$ energy to represent each 1, while the antipodal case would use $\mathcal{E}/v$ energy to represent each ± 1. The amplitude separation between the constellation points for the 0 and 1 in OOK is $\sqrt{\mathcal{E}/\omega}$, while the separation between -1 and 1 in antipodal is $2\sqrt{\mathcal{E}/v}$. Therefore, since it is desirable to have the separation between the constellation points as large as possible, we should choose OOK only when $\omega < v/4$. In the AND-ACCs presented in Section 5.4.1, the weight of each codevector is $\omega = v-k$. OOK is advantageous when $(3/4)v < k$, and antipodal modulation is preferable otherwise. Typically, in BIBDs with $\lambda = 1$ the block size k is much smaller than v [106] and therefore the antipodal form of code modulation is preferred.

If K colluders come together and average their marked content, then they produce an averaged test signal $\mathbf{y}$ whose contribution in the $\mathbf{u}$ component is the average of the b_{ij} values for that basis vector. In OOK, the b_{ij} are either 0 or 1 and therefore the values $0, 1/K, \cdots, (K-1)/K, 1$ are possible for the average $\overline{b}$ of the b_{ij} values for basis vector $\mathbf{u}$. Similarly, for the antipodal case, the values of $-1, -(K-2)/K, -(K-4)/K, \cdots, (K-4)/K, (K-2)/K, 1$ are possible. From these possibilities, it is clear that larger values of K are undesirable from a detection point-of-view. In the antipodal case, the separation between the $\overline{b} = (K-2)/K$ and $\overline{b} = 1$ (or $\overline{b} = (K-1)/K$ and $\overline{b} = 1$ for the OOK case) hypotheses is critical to the validity of using AND as the binary operation in designing an ACC. In order

to strengthen the validity of the AND assumption for a K-resilient AND-ACC, the separation between the $\bar{b} = (K-2)/K$ and $\bar{b} = 1$ hypotheses can be increased by devoting more energy $\mathcal{E}$ to the watermark, thereby operating at a higher WNR. Another possibility for improving the validity of the AND assumption is to increase the coding gain by using longer orthogonal basis vectors $\{\mathbf{u}_j\}$.

5.4.3 ACC Simulations with Gaussian Signals

In this section we study the behavior of our AND-ACC when used with code modulation in an abstract model, where $\mathbf{y}_j = \mathbf{x} + \mathbf{s}_j = \mathbf{x} + \alpha \sum_{i=1}^{v} b_{ij}\mathbf{u}_i$. The host signal $\mathbf{x}$ and the orthogonal basis signals $\mathbf{u}_i$ are assumed to be independent and each of them are vectors of i.i.d. Gaussian samples. The factor α is applied equally to all components and is used to control the WNR. We use these simulations to reveal the basic issues associated with collusion and code modulation. In the following subsection we present fingerprinting experiments using actual images.

In the simulations that follow, we used a $(16, 4, 1)$ BIBD to construct our AND-ACC code. The $(v, 4, 1)$ codes exist if and only if $v \equiv 1$ or $4 \pmod{12}$. We shall not describe how to construct this BIBD, but refer the reader to [106] for guidelines on constructing BIBDs. By complementing the incidence matrix, we get the code matrix presented in Table 5.1.

With this code, we use 16 orthogonal basis vectors to handle 20 users, and can uniquely identify up to $K = 3$ colluders.

We assumed that the host signal $\mathbf{x}$ was a $N = 10000$ point vector whose components were Gaussian $\mathcal{N}(0, 1)$. We generated a family of 16 random N-point vectors $\mathbf{u}$. The fingerprints for each user were assigned according to the antipodal form of code modulation, using the columns of $\mathbf{C}$ as the codevectors. For each user we

$$\mathbf{C} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1
\end{pmatrix}. \quad (5.17)$$

Table 5.1: Code matrix constructed from a $(16, 4, 1)$ BIBD.

embedded the fingerprints in $\mathbf{x}$ by adding a scaled version of the user's fingerprint to the host signal to produce a marked content signal $\mathbf{y}_j$. The scaling factor was applied equally to each component of the watermark, and was determined from the desired WNR $= 10 \log_{10} \|\mathbf{s}\|^2/\|\mathbf{x}\|^2$dB.

We first wanted to study the behavior of the detector and the legitimacy of the AND logic for the detector under the collusion scenario. We randomly selected 3 users as colluders and averaged their marked content signals to produce $\mathbf{y}_c$. The colluded content signal was correlated using T_N, as described in (5.4).

For three colluders, there are 4 possible values for $\bar{b}$, namely $-1, -1/3, 1/3$, and 1. We refer to the cases $-1, -1/3$ and $1/3$ as the non-1 hypothesis. We examined the tradeoff between the probability $p(1|1)$ of correctly detecting a 1 when a 1 was expected from the AND logic, and the probability of $p(1|\text{non-1})$, where the detector decides a 1 when the correct hypothesis was a non-1. The receiver operating characteristics (ROC) for a WNR of -25dB, -22.5dB, and -20dB were calculated and are presented in Figure 5.3. The ROC curve was generated using 2000 independent realizations of watermarking process for each threshold. There are a few interesting phenomena to point out. First, we observe that $p(1|1) = 1$ occurs at roughly $p(1|\text{non-1}) = 0.6$ for all three WNRs. This point corresponds to choosing a threshold $\tau = 0$, and indicates that the detector will falsely classify the $\bar{b} = 1/3$ case as a 1. However, this threshold appears to almost perfectly detect classify the true 1 cases.

We next calculated $p(1|1)$ and $p(1|\text{non-1})$ as a function of WNR for different thresholds. The thresholds used were $\tau_1 = 0.9E(T_N)$, $\tau_2 = 0.7E(T_N)$, and $\tau_3 = 0.5E(T_N)$. In order to calculate $E(T_N)$, we used (5.5) and assumed that the detector knows the the WNR and hence the power of the distortion. The plot of

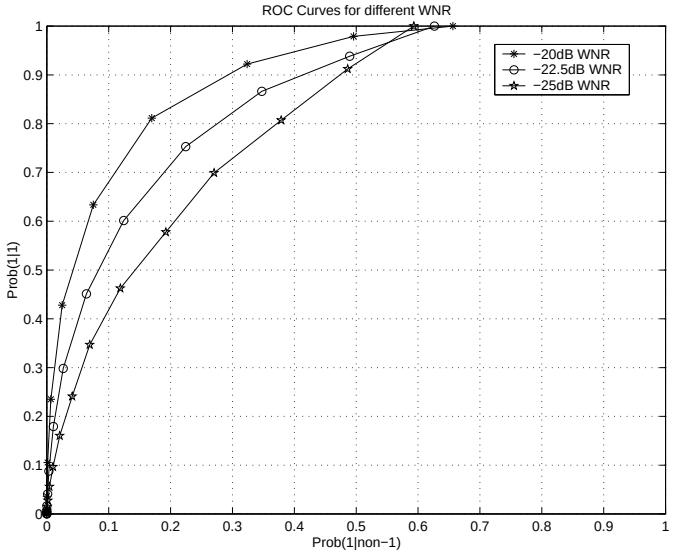
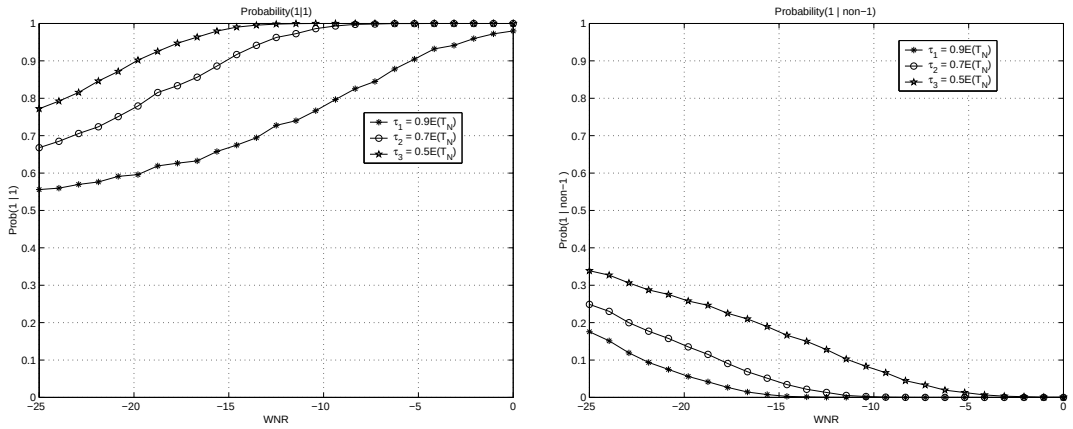


Figure 5.3: The receiver operating characteristic curve ($p(1|\text{non-1}$ vs. $p(1|1)$) for WNRs of -25dB, -22.5dB, and -20dB.

$p(1|1)$ for different threshold strategies is presented in Figure 5.4(a), and the plot of $p(1|\text{non-1})$ is presented in Figure 5.4(b). We observe that for the smaller threshold of $0.5E(T_N)$ the probability $p(1|1)$ is higher, but at an expense of a higher probability of false classification $p(1|\text{non-1})$. Increasing the threshold allows us to decrease the probability of falsely classifying a bit as a 1, but at an expense of also decreasing the probability of correctly classifying a bit as a 1.

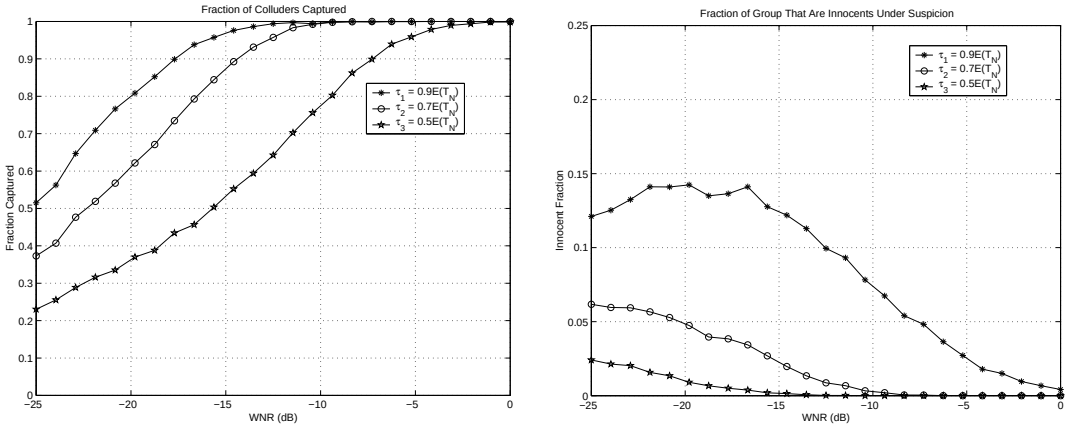
In order to see the effect that the properties of the detector have on identifying the colluders, we calculated the fraction of colluders that were captured as well as the fraction of the total group that were falsely placed under suspicion for different WNRs and different thresholds. In this experiment, we assumed that there were always 3 colluders, which were randomly selected from the entire user set. Using Algorithm 7, the locations of the 1s from the output of the detector were used to determine a set of users who are placed under suspicion. The fraction



(a) (b)

Figure 5.4: (a) The probability of detection $p(1|1)$ and (b) the probability of false alarm $p(1|\text{non-1})$ for different WNR and different thresholds.

of the colluders that belong to the suspicious set, and the fraction of the total user set that are innocents falsely placed under suspicion were calculated and averaged over 2000 realizations for each WNR. The results are presented in Figure 5.5. Compared to lower thresholds, for all WNRs, the higher threshold is able to capture more of the colluders, but also places more innocent users falsely under suspicion. As WNR increases, the detector has lower $p(1|\text{non-1})$, and therefore does not incorrectly eliminate colluders from suspicion. Similarly, at higher WNR, the detector has a higher $p(1|1)$, thereby correctly identifying more 1's, which allows for us to eliminate more innocents from suspicion. Therefore, at higher WNR we can capture more colluders as well as place less innocent users under suspicion. We note, however, that in Figure 5.5(b), at low WNR between -25dB and -15dB, the fraction of innocents under suspicion using threshold τ_1 is lower than at slightly higher WNR. This behavior can be explained by examining Figure 5.4(a) and Figure 5.4(b). We observe that at low WNR, the $p(1|\text{non-1})$ is higher



(a) (b)

Figure 5.5: (a) The fraction of colluders that are successfully captured, or placed under suspicion, and (b) the fraction of the total group that are innocent and falsely placed under suspicion for different WNR and different thresholds.

than slightly higher WNR, particularly for the threshold τ_1. However, for τ_1 the $p(1|1)$ at these WNR is relatively flat. These two observations combined indicate that at lower WNR we falsely decide 1 more often than at slightly higher WNR, while we do not experience much difference in the amount of correctly identified 1's. As more 1's pass through the detector we remove more users from suspicion. Therefore, since the amount of correctly detected 1's is roughly constant for WNRs between -25dB and -15dB, the additional 1's from false detections at lower WNR eliminates more innocent users (as well as colluders) from suspicion.

5.4.4 ACC Experiments with Images

In order to demonstrate the performance of our AND-ACC with code modulation fingerprinting on real images for fingerprinting users and detecting colluders, we used an additive spread spectrum watermarking scheme similar to that in [13],

User 1:	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	1	1	1
User 4:	-1	1	1	1	1	1	1	1	1	1	-1	-1	-1	1	1	1
User 8:	1	-1	1	1	1	1	-1	1	-1	1	1	1	1	1	-1	1
User(1,4) Average:	-1	0	0	0	1	1	1	1	1	1	0	0	0	1	1	1
User(1,4,8) Average:	$-\frac{1}{3}$	$-\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	1
After thresholding:	0	0	0	0	1	1	0	1	0	1	0	0	0	1	0	1

Table 5.2: The derived codevectors from a $(16, 4, 1)$ AND-ACC for user 1, user 4, and user 8. Also presented are the vectors from a two colluder scenario, and a three colluder scenario. The bottom row corresponds to the desired output of the detector using the AND logic for the three colluder case.

where the perceptually weighted watermark was added to 8×8 block DCT coefficients. The detection of the watermark is performed without the knowledge of the host image via the detection statistics as shown in (5.6). We used the same code matrix, given in Table 5.1, for the AND-ACC as in the simulations for Gaussian signals. The 512×512 Lenna and Baboon images were used as the host signals for the fingerprints. The fingerprinted images have no visible artifacts with an average PSNR of 41.2dB for Lenna, and 33.2dB for Baboon. Figure 5.6 shows the original images, the fingerprinted images, and the difference with respect to the originals.

The three derived code vectors that were assigned to user 1, 4, and 8 via antipodal mapping as well as the colluded versions are presented in Table 5.2. Two collusion examples are illustrated in Figure 5.7 and the detection statistics of the two examples are shown in Figure 5.8: In one example we averaged the Lenna images fingerprinted with user 1 and 4's codes, and the other is for averaging user 1, 4, and 8's. The colluded images are further compressed using JPEG with quality

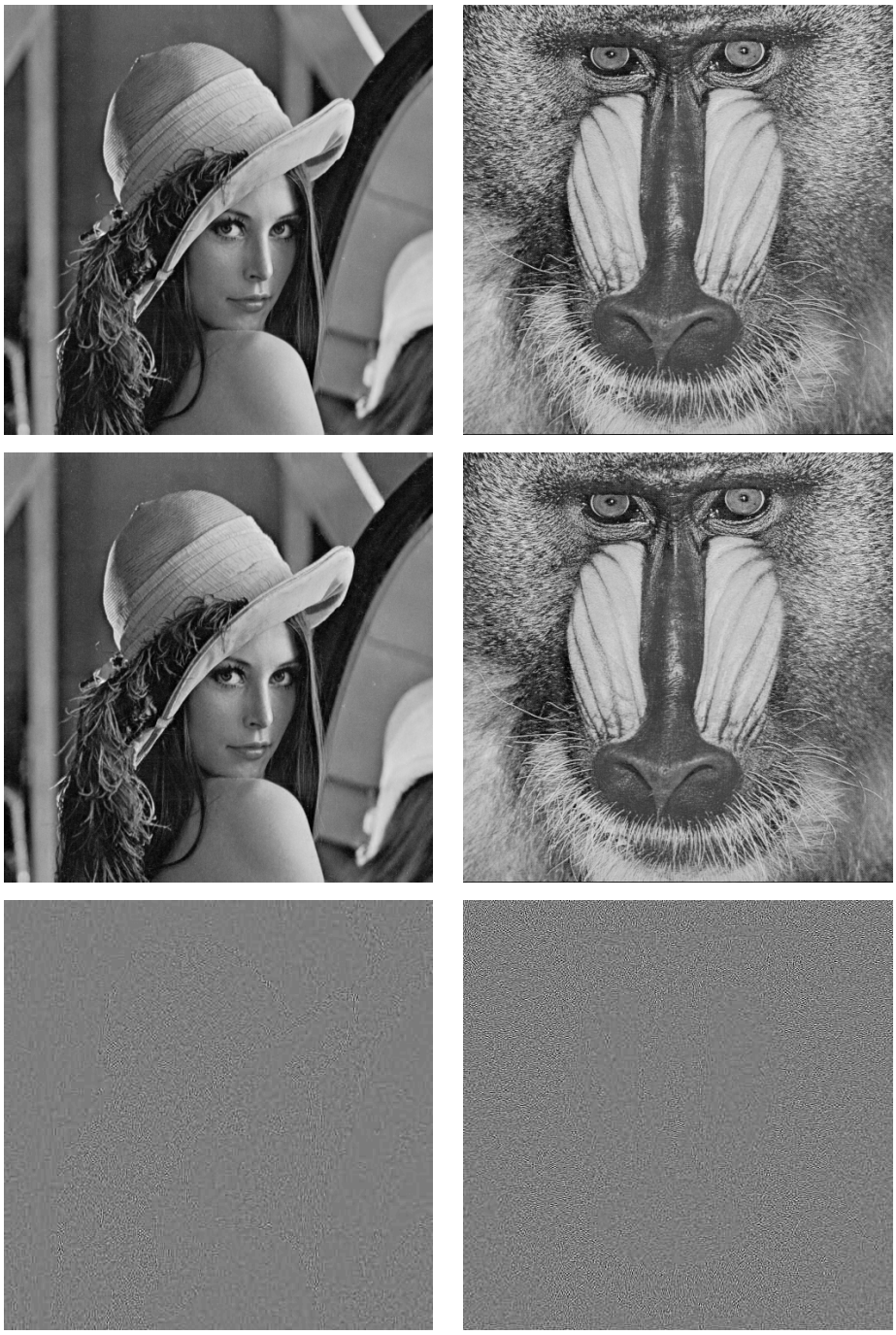


Figure 5.6: The original images (top), fingerprinted images (middle), and difference images (bottom) for Lenna and Baboon. In the difference images, gray color indicates zero difference between the original and the fingerprinted version, and brighter and darker indicates larger difference.

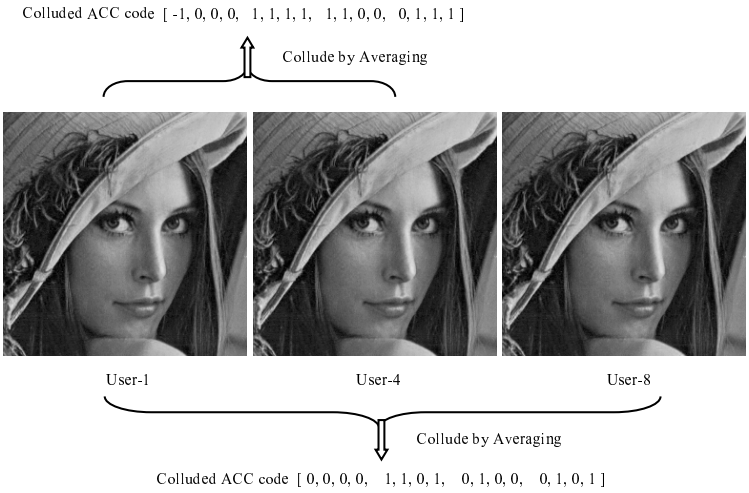


Figure 5.7: Illustration of collusion by averaging two and three images fingerprinted with ACC codes, respectively.

factor (QF) 50%. Also shown in Figure 5.8 are the thresholds determined from the estimated mean of the detection statistics $E(T_N)$. We then estimate the fingerprint codes by thresholding the detection statistics using a threshold of τ via

$$Th(x) = \begin{cases} +1 & \text{if } x \geq \tau \\ -1 & \text{if } x \leq -\tau \\ 0 & \text{otherwise} \end{cases} .$$

The threshold τ was calculated by scaling the estimated mean (5.7) with an empirical factor in the range of 0.5 to 0.8. The estimated fingerprint codes are identical to the expected ones shown in Table 5.2. We can see in Figure 5.8 and Figure 5.9 that non-blind detection increases the separation between the values of the detection statistics that are mapped to $\{-1, 0, +1\}$.

We present histograms of the T_N statistics from several collusion cases with different distortions applied to the colluded Lenna images in Figure 5.9. For each collusion and/or distortion scenario, we used 10 independent sets of basis vectors

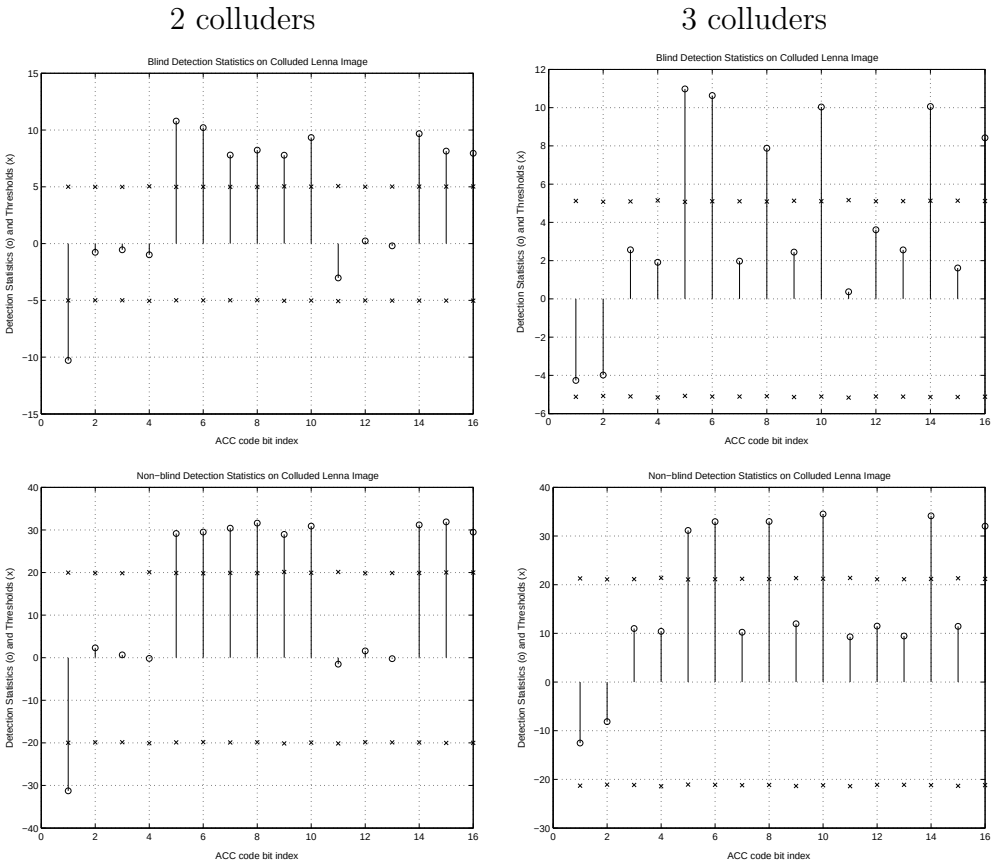


Figure 5.8: Example detection statistics values for 2 users' and 3 users' collusion with a $(16, 4, 1)$-BIBD AND-ACC fingerprint. (top) Blind detection scenario and (bottom) non-blind detection scenario. (left) User 1 and 4 perform averaging, resulting in the output of the detector as $(-1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1)$. (right) User 1, 4, and 8 perform averaging, resulting in the output of the detector as $(0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1)$.

to generate the fingerprints. Each set consists of 16 basis vectors for representing 16 ACC code bits. Figure 5.9 shows the histograms of the blind and non-blind detection scenarios, as well as the single user, two colluders and three colluders cases. We see that there is a clear distinction between the three decision regions. This implies that the average magnitude of T_N, when the bit values agree, is much larger than the average magnitude for where the bit values disagree, facilitating the accurate determination of the AND-ACC codes from colluded images. The separation of the three decision regions can also be observed from an aggregated histogram shown in Figure 5.10. The histogram depicts the blind detection statistics from a total of 8 collusion scenarios ranging from 1 to 3 colluders, and 4 distortion settings including no distortion, JPEG compression with quality factor of 50% and 90%, and low pass filtering.

Summarized in Table 5.3 are the bit error rates for the extracted fingerprint codes. The errors due to 1's being decoded as non-1's and non-1's being decoded as 1's are tallied separately. It should be noted that a significant portion of bit errors are caused by the inaccurate estimation of the mean of detection statistics, which directly affects the threshold setting, even though there is sufficient separation between the detection statistics corresponding to $\{-1, 0, +1\}$. This is especially the case for low bit rate JPEG compression and low pass filtering, where the assumption of i.i.d. Gaussian noise for estimating the mean detection statistics as described in (5.7) becomes less realistic. Improved detection statistics and mean estimation would reduce the bit errors, and in turn enhance the performance of fingerprinting.

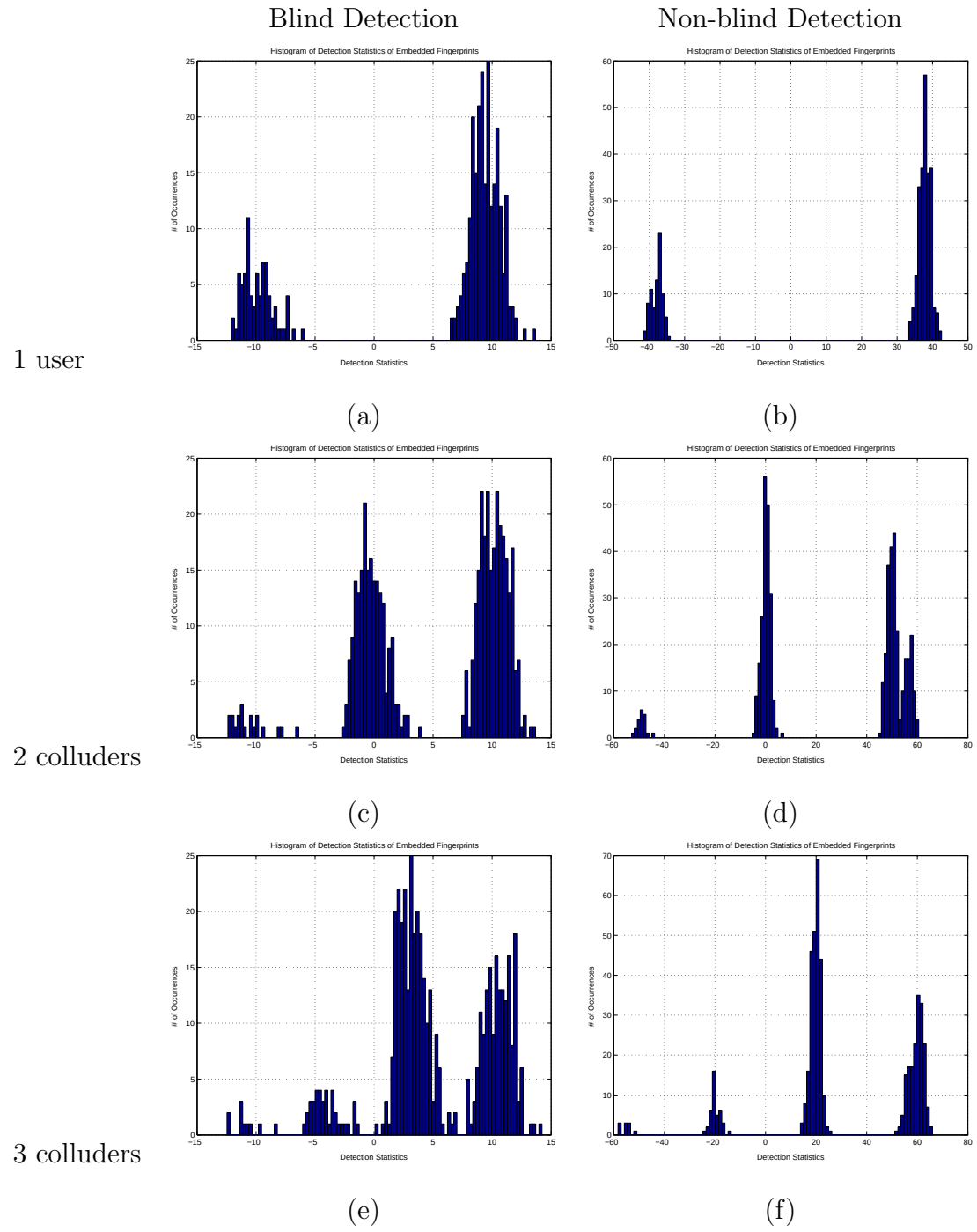


Figure 5.9: Histograms of detection statistics of embedded fingerprints: (top row) single fingerprint case, (middle row) 2-user collusion case, (bottom row) 3-user collusion case; (left column) blind detection, (right column) non-blind detection.

		no distortion	JPG 90%	JPG 50%	LPF
(non-1 $\leftarrow$ 1)	1-user	0	0	0	0.0375
	2-user	0	0	0	0.0115
	3-user	0	0	0	0.0056
(1 $\leftarrow$ non-1)	1-user	0	0	0	0
	2-user	0	0	0	0
	3-user	0.0667	0.0433	0.0433	0.0133

(a)

		no distortion	JPG 90%	JPG 50%	LPF
(non-1 $\leftarrow$ 1)	1-user	0	0	0.0042	0.0792
	2-user	0	0	0	0.0385
	3-user	0	0	0	0.0278
(1 $\leftarrow$ non-1)	1-user	0	0	0	0
	2-user	0	0	0	0
	3-user	0.0167	0.0167	0.01	0.01

(b)

Table 5.3: The bit error rate from the blind detector for (a) $\tau = 0.7E(T_N)$, and (b) $\tau = 0.8E(T_N)$.

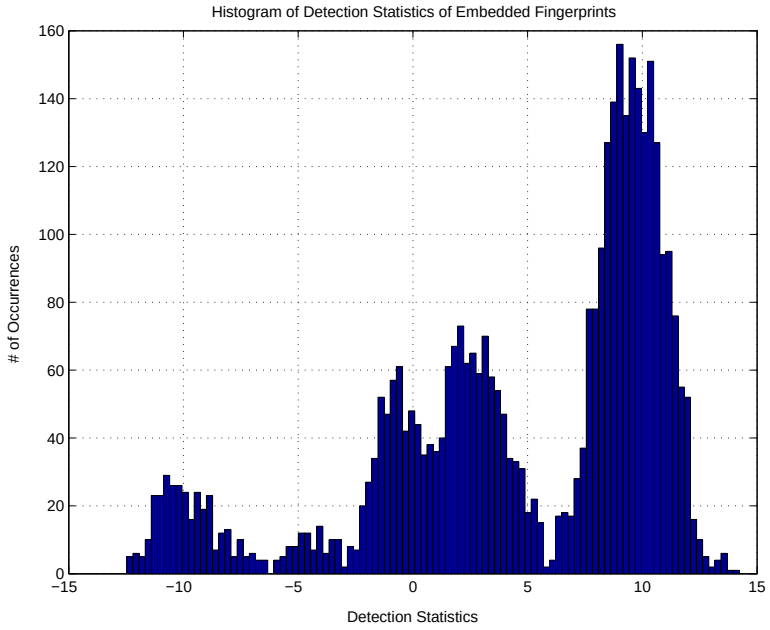


Figure 5.10: Aggregated histograms of blind detection statistics of embedded fingerprints covering from 1 to 3 colluders and 4 distortion settings.

5.5 Chapter Summary

In this chapter, we investigated the problem of making fingerprints for multimedia content that are resistant to collusion attacks. We introduced the collusion problem for additive embedding, and showed that under reasonable assumptions the optimal fair strategy for colluders performing an averaging attack is to perform an average where each user weighs their marked content equally.

We studied the effect that collusion has upon orthogonal, biorthogonal, and simplex modulation. The traditional detection schemes for orthogonal modulation in embedding applications require an amount of correlations that is linear in the amount of orthogonal signals. We presented a tree-based detection algorithm that reduces detection from linear to logarithmic complexity, and is able to efficiently

identify K colluders.

The drawback of orthogonal modulation for embedding is that it requires as many orthogonal signals as users. We developed a fingerprinting scheme based upon code modulation that does not require as many signals as orthogonal or simplex modulation in order to handle n users. We proposed anti-collusion codes (ACC) that are used in conjunction with modulation to fingerprint multimedia sources. Our anti-collusion codes have the property that the composition of any subset of K or fewer codevectors is unique, which allows for the identification of subgroups of K or fewer colluders. We constructed binary-valued ACC under the logical AND operation using combinatorial designs. Our construction is suitable for both the on-off keying and antipodal form of binary code modulation. Further, our codes are efficient in that they require only $\mathcal{O}(\sqrt{n})$ orthogonal signals to accommodate n users. For practical values of n this is an improvement over prior work on fingerprinting generic digital data.

We performed experiments to evaluate the proposed ACC-based fingerprints. We first used a Gaussian signal model to examine the ability of the ACC to identify the colluders, as well as reveal the amount of innocent users that would be falsely placed under suspicion. We observed a close interplay between the detector and the ability to capture colluders as well as the side-effect of placing innocent users under suspicion. We further observed that decreasing WNR increases the amount of false 1s that pass through the detector, which explains the small amount of colluders that are captured at low WNR. By raising the threshold, we improve the ability to capture colluders at all WNR, but also increase the amount of innocents who are falsely placed under suspicion. We also evaluated our fingerprints on real images, and observed that in both the blind and non-blind detection cases, the

values of the detection statistics were well-separated. This behavior allows the detector to accurately determine the colluders by correctly extracting a fingerprint codevector that corresponds to the colluder set.

Chapter 6

Conclusions

6.1 Thesis Summary

In this thesis we have examined the issue of multi-user security at three different stages of the media delivery and consumption process. In particular, the three stages that we focused on are the establishment of group keying material prior to content delivery, mechanisms that must occur during content delivery to address dynamic multicast group membership, and fingerprinting measures that must be in place to protect intellectual property following content delivery. Further, rather than treat security as a layer of a system that is separated from the application, we have examined issues related to the application or the network at each of the three stages that have an effect upon the design and operation of a secure application.

Prior to the delivery of group data, it is important to initially establish keying material used to secure the data in a group application. In Chapter 2, we proposed the butterfly scheme, a scalable method that constructs the conference key using the two-party Diffie-Hellman scheme as the basic building block. Underlying the butterfly scheme is a tree, called the conference tree, that describes the successive subgroups and subgroup keys that are formed en route to establishing the key for

the entire group. The use of tree-based conference key establishment reduces the amount of rounds needed to establish the conference key from being linear, as is the case in [4, 5], to being logarithmic in the group size.

The applications that will occur on the future networks will involve group members that are heterogeneous in terms of their computing power and communication capabilities. For applications with a diverse clientele, minimizing the total bandwidth or amount of rounds needed to form a group key might not be appropriate. Low-power devices, such as wireless appliances, cannot be expected to expend the same amount of computational effort as high-power devices, such as personal computers, when establishing a group secret. Therefore, the establishment of a conference key should address the varying user costs and resource constraints. Traditionally, conference keying schemes do not address this issue.

In order to address the different cost profiles that each user might have, in Chapter 2 we propose to tailor the design of the conference tree to account for different user costs. We assume that each user has a cost associated with performing one two-party DH scheme, or a budget that describes the amount of two-party DH schemes he is willing to participate in. We may seek conference trees with small average user cost by choosing the conference tree as the tree associated with coding a source with symbols whose weights are the different user costs. Source coding techniques, such as Huffman coding, allow for the design of conference trees that minimize the average user cost. In scenarios where the users have budget constraints the necessary conditions on being able to generate a tree-based conference key is that the vector of user budgets satisfy the Kraft Inequality. We also compared the heterogeneous tree-based scheme with other conference keying schemes by introducing the PESKY measure (the probability of establishing the session

key). PESKY quantifies the likelihood that a conference key can be established for a heterogeneous clientele of users whose budgets are drawn according to an underlying probability distribution. The PESKY of the proposed tree-based schemes was found to be larger than the PESKY for other schemes [107] when the user budget distribution did not have a single entity with significantly more resources than the other users.

Many applications that will occur will not have a static membership, but instead will have a dynamic membership, where users join and leave the service. The most appropriate framework for handling server-oriented content distribution with data confidentiality is by using a centralized entity that is responsible for maintaining the integrity of the users' keys. In secure multicast services, when users join and leave the group, it is necessary to update encryption keys in order to maintain the integrity of application's security.

Many schemes have been proposed to provide key management for server-based group applications, though the most common class of multicast key management schemes employ a tree hierarchy of keys [6]. The deficiency with the current multicast key management schemes that have been proposed is that their design has primarily focused on the issue of reducing the size of the payload (the rekeying information), and not on the size of the entire message (including the rekeying message and the header). In fact, the transmission of the messages that flag the users which portion of the message is intended for them can add significant communication overhead when used in conventional tree-based schemes.

In Chapter 3, we provided background and motivation for the development of a new message format for multicast key distribution. This new message format, which we call the *residue-based* format is constructed using one-way functions and

large integer arithmetic and provides a single *homogenized* message from which each user can extract the new key. The advantage of the residue-based format lies in the fact it does not require the usage of header information to flag the users to their portion of the message. We then provide an analysis of the security of the message format, and explain the motivation for using one-way functions to achieve reusability in the private user keys.

The residue-based message format, or other homogenized message formats, may be used in conjunction with a tree-based key hierarchy to reduce the communication overhead associated with rekeying and achieve desirable communication scalability. In Chapter 4, we proposed a multicast key management system that uses a homogenized message format that does not require any additional communication overhead in order for users to access their rekeying information. We employed a message format based upon polynomial interpolation [9], where a single message consists of information needed to build a polynomial from which each user can calculate the new key. Compared with the traditional format of the rekeying messages used in tree-based multicast key management, where the rekeying message consists of the concatenation of messages intended for different subgroups, our composite message format reduces the amount of header information, while maintaining the same payload size. Further, we optimize the parameters associated with the design of the tree by introducing a stochastic population model where each leaf node is occupied according to i.i.d. Bernoulli random variables. We showed that, under this occupancy model, the average case communication requirements are close to the worst-case communication requirements. Therefore, we may optimize the tree for the worst-case and not suffer any significant performance loss.

Mechanisms that provide data confidentiality or access control only protect

the content while it remains in the protected, or encrypted domain. Following decryption, it is possible for users to access cleartext representations of the content. These cleartext representations may then be redistributed, affecting the digital rights of the content owners and media distributors. Digital fingerprints, which remain after content decryption, are a powerful technique to control the redistribution of content, and allow for the possibility of tracing the consumers who use their content for unintended purposes. Digital fingerprinting for multimedia sources is accomplished through data embedding, or watermarking techniques.

A cost-efficient attack against multimedia fingerprinting can be waged by a coalition of users with the same content that contains different marks. One of the simplest approaches to performing such a *collusion* attack is to average multiple copies of the content together. Other collusion attacks might involve forming a new content by selecting different pixels or blocks from the different colluders' content. By gathering a large enough coalition of colluders, it is possible to sufficiently attenuate each of the colluders' identifying fingerprints and produce a new version of the content with no detectable fingerprints.

In Chapter 5, we designed fingerprints for multimedia content, such as images and video, that are resistant to collusion attacks. We study the effect that collusion has upon the constellation points in orthogonal and simplex modulation, and present an efficient tree-based detection scheme that is able to identify a set of colluders with an amount of correlations that is logarithmic in the number of basis vectors. This is a significant improvement over the linear complexity traditionally associated with identifying orthogonal fingerprints. We then developed a fingerprinting scheme based upon code modulation that requires only $\mathcal{O}(\sqrt{n})$ orthogonal signals to accommodate n users. We proposed anti-collusion codes (ACC) that are

used in conjunction with code modulation to fingerprint multimedia sources. We constructed binary-valued ACC under the logical AND operation using combinatorial designs. We then performed experiments using a Gaussian signal model to examine the ability of the ACC to identify the colluders, as well as reveal the amount of innocent users that would be falsely placed under suspicion. Further, we applied ACC-based fingerprints to real images, and observed separation of the detector hypotheses in the detection statistics. This behavior allows the detector to accurately determine the colluders by correctly extracting a fingerprint codevector that corresponds to the colluder set.

6.2 Future Work

The world is rapidly evolving and, as people are brought closer together, there will be an increased need for security and digital rights solutions across a broad range of applications. There are many possible directions for growth in the field of multi-user security.

As wireless networks truly become ubiquitous, consumers will demand a suite of multimedia applications for their low-powered, wireless devices. It will be necessary to introduce scalable security solutions, much like scalable coding provides a scalable solution for source coding, in order to accommodate users with less computational capabilities. Contemporary encryption technologies treat data as either a block or a stream of bits and usually do not make use of the inherent structures or the syntax of the data. Applying encryption directly to the bit stream is not appropriate for multimedia services because of the vast quantity of data to be processed. Many multimedia services, such as pay-per-view and video-on-demand, require the real-time encryption of a huge amount of data. Encrypting and de-

crypting the vast quantities of data associated with multimedia sources requires significant processing resources that may not be available, especially for mobile and portable devices. It may be possible to address these concerns by reducing the amount of computational resources needed through a suite of different encryption algorithms of varying power and expense. Stronger and more expensive algorithms would be used to protect more critical portions of the scalable bit stream, while less powerful algorithms would be used to protect less important layers of the media. The challenge here lies in determining an appropriate tradeoff between desired levels of data confidentiality and the computational budget constraints.

Further, many multimedia applications will involve data that is distributed in multi-layered or multi-object format. For example, in an HDTV broadcast, users with a normal TV receiver can receive the normal format, while other users with an HDTV receiver can receive both the normal format and the extra information needed to achieve HDTV resolution. As another example, the MPEG-4 standard allows for the composition of multiple media streams corresponding to different object planes [45]. In both of these cases, it will be desirable for service providers to have efficient schemes for maintaining access control to the different layers of media. Traditional multicast key management schemes are not designed to handle the key management issues associated with multiple services occurring concurrently that have correlated memberships. Multi-layered or multi-object services, as is prevalent in multimedia applications, will consist of users that subscribe to different objects or layers. New key management schemes should be designed that exploit the overlap in the memberships of the different objects or services, while incorporating new functionalities that are not present in conventional multicast key management schemes. 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. Preliminary work into this problem was presented in [14, 76, 108].

There is further work that can be done to provide collusion-resistant fingerprinting for multimedia content. The current solutions to this problem, which were presented in Chapter 5, have only scratched the surface of the interplay between the detector and the coding aspect of the fingerprint's design. One direction for further exploration is to consider the design of ACC for more general detector logics. Currently, our construction assumes that the detector treats the collusion of a set of bits as the logical AND of those bits. In the data embedding scenario, this assumption is only an approximation that becomes less valid as the amount of colluders increases. A more sophisticated approach for joint detector and code design would use soft-decision decoding, which would allow for the design of an ACC that can handle larger colluder sets. Another avenue of exploration would be in the design of the fingerprints themselves. Currently, the fingerprints are built by using binary-valued codes in conjunction with code modulation. This means that the fingerprints are constructed from linear combinations of orthogonal basis signals where each coefficient in the expansion has only two possible values. Due to this restriction, the code vectors of our ACC codes are forced to be the vertices of a multidimensional hypercube. A further direction for examination would be to examine the possibility of using real-valued coefficients in the expansion. The relaxation of the coefficients to be real-valued would allow the code vectors to lie on a unit sphere that circumscribes the hypercube. Since the sphere contains the hypercube, it might be possible to form more code vectors for a given amount of orthogonal waveforms, and hence allow for us to accommodate more users.

In addition to the directions for security solutions for multimedia applications, there are many possibilities to apply networking techniques to develop a security framework for more general multicast applications consisting of heterogeneous clientele. To address security for the generic heterogeneous applications, we may divide the users into different classes that would be assigned to different multicast groups and treated independently of each other by the network. Multicast routing protocols would build different routing trees for each user class, and thus the necessary rekeying messages for different classes of users would be sent via separate routes. Changes in the membership for one class of users would affect the routing tree for that class only, and not require that other classes update their multicast tree. Additionally, the use of different multicast groups might increase the reliability of the delivery of rekeying messages.

The future will be an exciting time to perform research in the area of security and digital rights management. The rapid advancements in technology promises to create many opportunities to build security solutions through combining techniques from a broad array of fields, such as cryptography, communications, networking, and signal processing.

BIBLIOGRAPHY

[1] "Secure digital music initiative," See http://www.sdmi.org.

[2] ISO/IEC JTC1 SC29 WG11 N2614, "MPEG-4 intellectual property management and protection (IPMP) overview and applications document," http://www.cselt.it/mpeg/public/mpeg-4_ipmp.zip, December 1998.

[3] ISO/IEC JTC1/SC29/WG11 N3939, "MPEG-21 proposed draft technical report," http://www.cselt.it/mpeg/public/mpeg-21_pdtr.zip, January 2001.

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

[5] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman key distribution extended to group communication," in *Proc. 3rd ACM Conf. on Computer Commun. Security*, 1996, pp. 31–37.

[6] 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.

[7] 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.

[8] R. Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas, "Multicast security: a taxonomy and some efficient constructions," in *IEEE INFOCOM'99*, 1999, pp. 708 –716.

[9] M. Just, E. Kranakis, D. Krizanc, and P. vanOorschot, "On key distribution via true broadcasting," in *Proc. 2nd ACM Conf. on Computer and Communications Security*, 1994, pp. 81–88.

[10] B. Chor, A. Fiat, M. Naor, and B. Pinkas, "Tracing traitors," *IEEE Tran. on Information Theory*, vol. 46, pp. 893–910, May 2000.

[11] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Tran. on Information Theory*, vol. 44, pp. 1897–1905, September 1998.

[12] I. Cox, J. Kilian, F. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Tran. on Image Proc.*, vol. 6(12), pp. 1673–1687, December 1997.

[13] C. Podilchuk and W. Zeng, "Image adaptive watermarking using visual models," *IEEE Journal on Selected Areas in Communications*, vol. 16(4), pp. 525–540, May 1998.

[14] W. Trappe, J. Song, R. Poovendran, and K.J.R. Liu, "Key distribution for secure multimedia multicasts via data embedding," in *IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, 2001.

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

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

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

[18] G. Ateniese, M. Steiner, and G. Tsudik, "New multiparty authentication services and key agreement protocols," *IEEE Journal on Selected Areas of Communications*, vol. 18, pp. 628 –639, 2000.

[19] M. Steiner, G. Tsudik, and M. Waidner, "Key agreement in dynamic peer groups," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, pp. 769 –780, 2000.

[20] V. Miller, "Use of elliptic curves in cryptography," *Advances in Cryptology: Crypto '85*, pp. 417–426, 1986.

[21] W. Trappe, Y. Wang, and K.J.R. Liu, "Group key agreement using divide-and-conquer strategies," in *Conference on Information Sciences and Systems, The John's Hopkins University*, March 2001.

[22] W. Trappe, Y. Wang, and K.J.R. Liu, "Establishment of conference keys in heterogenous networks," in *IEEE Int. Conference on Communications*, 2002, (Accepted, available at http://www.eng.umd.edu/~wxt/papers/grkey_icc2002.pdf).

[23] A. Oppenheim and R. Schafer, *Discrete-time Signal Processing*, Prentice Hall, 1989.

[24] T. Cover and J. Thomas, *Elements of Information Theory*, John Wiley and Sons, 1991.

[25] D. Huffman, "A method for the construction of minimum redundancy codes," *Proc. of IRE*, vol. 40, pp. 1098–1101, 1952.

[26] D. A. Lelewer and D. S. Hirschberg, "Data compression," *ACM Computing Surveys*, vol. 19, pp. 261–296, 1987.

[27] D. E. Knuth, *The Art of Computer Programming, Vol. 3, Sorting and Searching*, Addison Wesley, 1973.

[28] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, Mc. Graw Hill, 1998.

[29] A. Turping and A. Moffat, "Practical length-limited coding for large alphabets," *Computer Journal*, vol. 38, pp. 339–347, 1995.

[30] D. C. Van Voorhis, "Constructing codes with bounded codeword lengths," *IEEE Transactions on Information Theory*, vol. 20, pp. 288–290, March 1974.

[31] L. Larmore and D. Hirschberg, "A fast algorithm for optimal length-limited Huffman codes," *Journal of the ACM*, vol. 37, pp. 464–473, July 1990.

[32] R. Milidiu and E. Laber, "The warm-up algorithm: a Lagrangian construction of length restricted Huffman codes," *SIAM Journal of Computation*, vol. 30, pp. 1405–1426, 2000.

[33] M. R. Garey, "Optimal binary search trees with restricted maximal depth," *SIAM Journal of Computing*, vol. 3, pp. 101–110, June 1974.

[34] E. Gilbert, "Codes based on inaccurate source probabilities," *IEEE Trans. on Inform. Theory*, vol. 17, pp. 304–314, 1971.

[35] H. Murakami, S. Matsumoto, and H. Yamamoto, "Algorithm for construction of variable length code with limited maximum word length," *IEEE Transactions on Communications*, vol. 32, pp. 1157–1159, Oct. 1984.

[36] B. Fox, "Discrete optimization via marginal analysis," *Management Science*, vol. 13, pp. 210–216, 1966.

[37] L. A. Wolsey, *Integer Programming*, John Wiley and Sons, 1998.

[38] T. C. Hu, *Integer Programming and Network Flows*, Addison Wesley, 1969.

[39] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley and Sons, 1999.

[40] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," *Econometrica*, vol. 28, pp. 497–520, 1960.

[41] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations Research*, vol. 14, pp. 699–719, 1966.

[42] D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An algorithm for the traveling salesman problem," *Operations Research*, vol. 11, pp. 972–989, 1963.

[43] T. Nemetz, "On the word-length of Huffman codes," *Probl. Contr. and Inform. Theory*, vol. 9, pp. 231–242, 1980.

[44] F. Fabris, A. Sgarro, and R. Pauletti, "Tunstall adaptive coding and miscoding," *IEEE Trans. on Inform. Theory*, vol. 42, pp. 2167–2180, 1996.

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

[46] J. Chen, U. Koc, and K. J. R. Liu, *Design of Digital Video Coding Systems*, Marcel Dekker Inc., 2002.

[47] K. Ngan, C. Yap, and K. Tan, *Video Coding for Wireless Communication Systems*, Marcel Dekker Inc., 2001.

[48] M. Sun and A. Reibman, *Compressed Video over Networks*, Marcel Dekker Inc., 2001.

[49] R. Canetti, T. Malkin, and K. Nissim, "Efficient communication-storage tradeoffs for multicast encryption," *Eurocrypt*, pp. 456–470, 1999.

[50] F. Hartung and B. Girod, "Digital watermarking of MPEG-2 coded video in the bitstream domain," *IEEE Int. Conf. Accostic Speech and Signal Proc. '97*, pp. 2621–2624, 1997.

[51] M. Wu and B. Liu, "Modulation and multiplexing techniques for multimedia data hiding," in *Proc. of SPIE ITcom'01, SPIE vol 4518*, Aug. 2001.

[52] C. Blundo, L.A. Frota Mattos, and D. R. Stinson, "Multiple key distribution maintaining user anonymity via broadcast channels," *J. Computer Security*, vol. 3, pp. 309–323, 1994.

[53] S. Paul, *Multicasting on the Internet and its Applications*, Kluwer Academic, 1998.

[54] D. Balenson, D. McGrew, and A. Sherman, "Key management for large dynamic groups: one-way function trees and amortized initialization," Internet Draft Report.

[55] R. Poovendran and J.S. Baras, "An information theoretic approach for design and analysis of rooted tree-based multicast key management schemes," *Advances in Cryptology: Crypto '99*, pp. 624–638, 1999.

[56] U. Horn, K. Stuhlmller, M. Link, and B. Girod, "Robust internet video transmission based on scalable coding and unequal error protection," *Image Communication*, vol. 15, pp. 77–94, Sept 1999.

[57] H. Zheng and K.J.R. Liu, "Optimization approaches for delivering multimedia services over digital subscriber lines," *IEEE Signal Processing Magazine*, vol. 17, pp. 44–60, July 2000.

[58] W. Trappe and L.C. Washington, *Introduction to Cryptography with Coding Theory*, Prentice Hall, 2002.

[59] C. Herpel, A. Eleftheriadis, and G. Franceschini, "MPEG-4 systems: elementary stream management and delivery," in *Multimedia Systems, Standards, and Networks*, A. Puri and T. Chen, Eds., pp. 367–405. Marcel Dekker Inc., 2000.

[60] J. Mitchell, W. Pennebaker, C. Fogg, and D. LeGall, *MPEG Video Compression Standard*, Chapman & Hall, 1997.

[61] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 784–803, 1997.

[62] J. Lin and S. Paul, "RMTP: A reliable multicast transport protocol," in *INFOCOM*, San Francisco, CA, Mar 1996, pp. 1414–1424.

[63] S. Paul, K. K. Sabnani, J. Lin, and S. Bhattacharyya, "Reliable multicast transport protocol (RMTP)," *IEEE Journal of Selected Areas in Communications*, vol. 15, pp. 407–421, 1997.

[64] R. Poovendran and J.S. Baras, "An information-theoretic approach for design and analysis of rooted-tree-based multicast key management schemes," *IEEE Trans. on Information Theory*, vol. 47, pp. 2824 –2834, 2001.

[65] J. Song, R. Poovendran, W. Trappe, and K.J.R. Liu, "A dynamic key distribution scheme using data embedding for secure multimedia multicast," in *Proceedings of SPIE 2001 Security and Watermarking for Multimedia*, San Jose, CA, 2001.

[66] A. Westfeld and G. Wolf, "Steganography in a video conferencing system," in *Proc. 2nd International Workshop on Information Hiding*, 1998, pp. 32–47.

[67] J. Song and K. J. R. Liu, "A data embedding scheme for H.263 compatible video coding," *IEEE ISCAS*, vol. 4, pp. 390–393, June 1999.

[68] J. Song and K. J. R. Liu, "A data embedded video coding scheme for error-prone channels," *IEEE Trans. on Multimedia*, vol. 3, pp. 415–423, Dec. 2001.

[69] A. Menezes, P. vanOorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[70] M. Wu, *Multimedia Data Hiding*, Ph.D. thesis, Princeton University, 2001.

[71] J. Nonnenmacher, E. W. Biersack, and D. Towsley, "Parity-based loss recovery for reliable multicast transmission," *IEEE/ACM Trans. Networking*, vol. 5, pp. 349–361, Aug 1998.

[72] D. Stinson, *Cryptography: Theory and Practice*, CRC Press, 1995.

[73] J. B. Conway, *Functions of One Complex Variable*, Springer-Verlag, 2nd edition, 1978.

[74] ITU-T Rec. H263, "Version 2, video coding for low bitrate communication," Jan. 1998.

[75] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*, Addison Wesley, 1997.

[76] W. Trappe, J. Song, R. Poovendran, and K.J.R. Liu, "A dynamic key distribution scheme using data embedding for secure multimedia multicast," *Submitted to IEEE Trans. on Multimedia*, 2000.

[77] H. Eriksson, "Mbone: The multicast backbone," *Communications of the ACM*, vol. 37, pp. 54–60, August 1994.

[78] H. Harney and C. Muckenhirn, "Gkmp specification," Internet Request for Comments 2094, July 1997.

[79] A. Fiat and M. Naor, "Broadcast encryption," *Advances in Cryptology: Crypto '93*, pp. 480–491, 1993.

[80] R. Poovendran, "A convex optimization approach for addressing storage-communication tradeoffs in multicast encryption," Tech. Rep. CS-TR-4082, Department of Computer Science, University of Maryland, 1999.

[81] J. Daemen and V. Rijmen, "AES proposal: Rijndael," See http://crsc.nist.gov/encryption/aes, 2000.

[82] ITU-T Recommendation X.509 (1997), "The director: Authentication framework," 1997.

[83] N. Koblitz, *A Course in Number Theory and Cryptography*, Springer-Verlag, 2nd edition, 1994.

[84] K. Atkinson, *An Introduction to Numerical Analysis*, John Wiley & Sons, 2nd edition, 1989.

[85] G. Golub and C. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 3rd edition, 1996.

[86] N. Heintze, "Scalable document fingerprinting," in *1996 USENIX Workshop on Electronic Commerce*, November 1996.

[87] H. S. Stone, "Analysis of attacks on image watermarks with randomized coefficients," *NEC Technical Report 96-045*, 1996.

[88] S. Craver, N.Memon, B-L. Yeo, and M.M. Yeung, "Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks, and implications," *IEEE Journal on Selected Areas in Communication*, vol. 16, pp. 573–586, May 1998.

[89] W. Zeng and B. Liu, "A statistical watermark detection technique without using original images for resolving rightful ownerships of digital images," *IEEE Tran. on Image Proc.*, vol. 8(11), pp. 1534–1548, November 1999.

[90] B. Chen and G.W. Wornell, "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *IEEE Trans. on Info. Theory*, vol. 47, pp. 1423–1443, May 2001.

[91] H. V. Poor, *An Introduction to Signal Detection and Estimation*, Springer Verlag, 2nd edition, 1994.

[92] M. Wu, H. Yu, and A. Gelman, "Multi-level data hiding for digital image and video," in *Proceedings of SPIE, Photonics East Conference on Multimedia Systems and Applications*, 1999, vol. 3845.

[93] J. G. Proakis, *Digital Communications*, McGraw-Hill, 3rd edition, 1995.

[94] A. Herrigel, J. Oruanaidh, H. Petersen, S. Pereira, and T. Pun, "Secure copyright protection techniques for digital images," in *Second Information Hiding Workshop (IHW), Lecture Notes in Computer Science*, vol. 1525. Springer-Verlag, 1998.

[95] M. Sobel and P. Groll, "Binomial group-testing with an unknown proportion of defectives," *Technometrics*, vol. 8, pp. 631–656, Nov. 1966.

[96] D. Z. Du, G. L. Xue, S. Z. Sun, and S. W. Cheng, "Modifications of competitive group testing," *SIAM Journal of Computing*, vol. 23, pp. 82–96, Feb. 1994.

[97] D. Z. Du and H. Park, "On competitive group testing," *SIAM Journal of Computing*, vol. 23, pp. 1019–1025, Oct. 1994.

[98] D. Z. Du and F. K. Hwang, *Combinatorial Group Testing and Its Applications*, World Scientific Publishing Co., 2000.

[99] J. Aslam and A. Dhagat, "Searching in the presence of linearly bounded errors," in *Proceedings of the 23rd ACM Symposium on Theory of Coputing*, May 1991, pp. 486–493.

[100] J. Dittmann, P. Schmitt, E. Saar, J. Schwenk, and J. Ueberberg, "Combining digital watermarks and collusion secure fingerprints for digital images," *SPIE Journal of Electronic Imaging*, vol. 9, pp. 456–467, 2000.

[101] T. W. Hungerford, *Algebra*, Springer-Verlag, 1974.

[102] J. H. Dinitz and D. R. Stinson, *Contemporary Design Theory: A Collection of Surveys*, John Wiley and Sons, 1992.

[103] I. Anderson, *Combinatorial Designs and Tournaments*, Oxford University Press, 1997.

[104] C.C. Lindner and C.A. Rodger, *Design Theory*, CRC Press, 1997.

[105] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications*, Cambridge Press, 1994.

[106] C. J. Colbourn and J. H. Dinitz, *The CRC Handbook of Combinatorial Designs*, CRC Press, 1996.

[107] W. Trappe, Y. Wang, and K.J.R. Liu, "Conference key establishment for heterogeneous networks," *Submitted to IEEE/ACM Trans. on Networking*, 2002.

[108] W. Trappe, J. Song, R. Poovendran, and K.J.R. Liu, "Dynamic M^4: A dynamic multicast key management scheme for groups of mobile multimedia users," *Presented at MPEG-4 IPMP Meeting in La Baule, France*, 2000.