Key Management and Distribution for Secure Multimedia Multicast

Wade Trappe, *Member, IEEE*, Jie Song, Radha Poovendran, *Member, IEEE*, and K. J. Ray Liu, *Fellow, IEEE*

***Abstract*—The problem of controlling access to multimedia multicasts requires the distribution and maintenance of keying information. Typically, the problem of key management is considered separately from the problem of distributing the rekeying messages. Multimedia sources provide two approaches to distributing the rekeying messages associated with securing group communication. The first, and more conventional, approach employs the use of a media-independent channel to convey rekeying messages. We propose, however, a second approach that involves the use of a media-dependent channel, and is achieved for multimedia by using data embedding techniques. Compared to a media-independent channel, the use of data embedding to convey rekeying messages provides enhanced security by masking the presence of rekeying operations. This covert communication makes it difficult for an adversary to gather information regarding the group membership and its dynamics. In addition to proposing a new mode of conveyance for the rekeying messages, we introduce a new message format that is suitable for multicast key management schemes. This new message format 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, as an example of the interplay between the key management scheme and the mode of conveyance, we study 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. Finally, since multimedia services will involve multiple layers or objects, we extend the tree-based key management schemes to include new operations needed to handle multilayer multimedia applications where group members may subscribe or cancel membership to some layers while maintaining membership to other layers.**

***Index Terms*—Data embedding, key management, multimedia, secure multicast.**

Manuscript received October 26, 2000; revised July 3, 2002. The work of W. Trappe, J. Song, and K. J. R. Liu was supported in part by the NSF NYI Award MIP9457397. The work of R. Poovendran was supported in part by the NSF CAREER Grant ANI-0093187. The associate editor coordinating the review of this paper and approving it for publication was Dr. Minerva Yeung.

W. Trappe is with the Wireless Information Network Laboratory and the Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, Piscataway, NJ 08854 USA (e-mail: trappe@winlab.rutgers.edu).

J. Song is with the Agere Systems, Holmdel, NJ 07733 (e-mail: jiesong@agere.com).

R. Poovendran is with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195-2500 USA (e-mail: radha@ee.washington.edu).

K. J. R. Liu is with the Department of Electrical and Computer Engineering and the Institute for Systems Research, University of Maryland, College Park, MD 20742-0001 USA (e-mail: kjrliu@eng.umd.edu).

Digital Object Identifier 10.1109/TMM.2003.813279

I. 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. 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 nonprivileged users.

The problem of 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. Both of these scenarios might arise in conferences where prior and future information might be confidential and meant for select subgroups. Furthermore, any solution to access control should address issues of resource scalability for scenarios consisting of large privileged groups.

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 [1]–[4]. 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 [5]–[8] and operate in a scalable or layered format [9], which we can exploit to achieve an improved design of an access control system for multimedia multicasts.

In this paper we study the problem of key management for multimedia multicast services. We begin in Section II

by introducing the concepts of key management in multicast communication and present a basic key management scheme that will be used later in the paper in Section III. In Section IV, 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 V, 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 VI, 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 VII, 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 VIII.

II. Key Management in Multicast Communications

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. The efficiency in multicast communication has created many new application areas, and made others more feasible [10]. For the commercial success of most of these applications, it is essential to control access to the data. Only 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 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.

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 order to motivate the importance of scalability, we present the simplest 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 two 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 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 by Wallner *et al.* [3], and independently by Wong *et al.* [1]. 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 $\mathcal{O}(\log n)$. We note that the $\mathcal{O}$ notation is presented to indicate that the constant factors are implementation dependent. Various modifications to the tree-based key management scheme have been proposed to improve the usage of communication and storage resources. For example, in [4] the tradeoffs between storage and communication requirements are studied, and a modification to the tree-based schemes of [1], [3] is presented that achieves sublinear server-side storage. Further, in [11], 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.

III. A Basic Key Management Scheme

In this section, we present a simplified key management scheme that will be used in the discussions in Section V-A, 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.

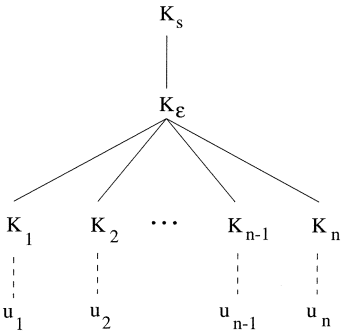


Fig. 1. Basic key distribution scheme.

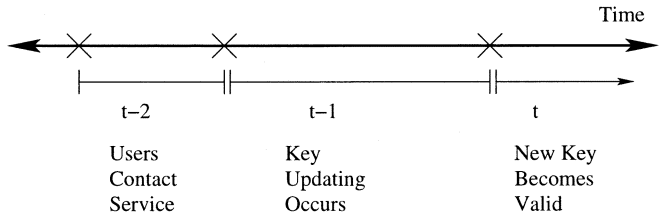


Fig. 2. Time intervals $t-2$, $t-1$ and t used in the paper. The joining/departing user contacts the service during time interval $t-2$, the rekeying messages are transmitted during $t-1$, and new key information takes effect at the beginning of time interval t.

Consider a group of n multimedia users who will share a multimedia multicast. In the simple key distribution scheme for n users, depicted in Fig. 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 $K_1, K_2, \ldots, K_n$ 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 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 Fig. 2. Observe that it is not necessary that the time intervals have the same duration.

A. 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.

B. 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.

C. 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

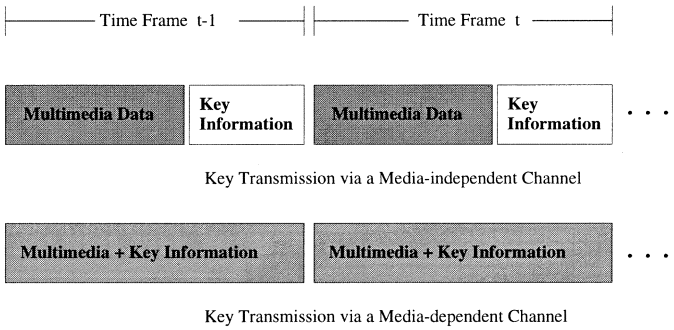


Fig. 3. Two approaches to distributing the key information in multimedia multicasting: Using a media-independent channel, and using a media-dependent channel.

the GC forms a new key $K_\epsilon(t)$ and encrypts it with the keys K_j for $j \neq n$. A single message

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

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.

IV. 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 Fig. 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.

In a conventional, nonsecure 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 [9]. 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) [12], [13].

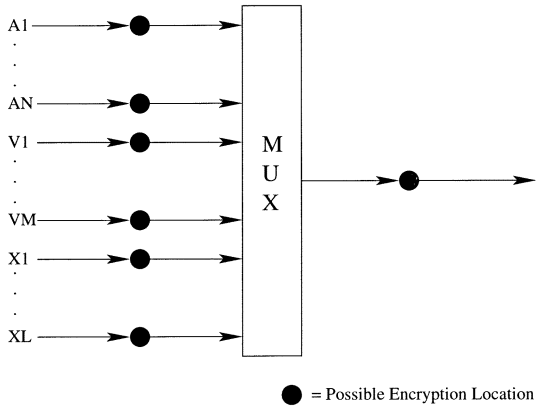


Fig. 4. A generic multiplexing diagram depicting several audio streams $(A1–AN)$, video streams $(V1–VM)$, and auxiliary streams $(X1–XL)$. Also depicted are locations where encryption is possible.

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 Fig. 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 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 [14], 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.[1]

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.

[1]In fact, in the MPEG-4 IPMP framework IPMP control points are located prior to the DMIF at the encoder [15].

A. 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 13 818-1) 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 [15]. 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 nonnormative application-specific parameters. For example, in MPEG-1 video, the bitstream format for the video sequence layer, the group of pictures layer, and the picture layer provides a mechanism to convey optional *user* data [16]. 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 [10]. 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 acknowledgment message from each of the clients announcing that they have successfully received the rekeying message. Further, the server should drop users from the service who do not acknowledge the receipt of the rekeying message after several retransmissions in order to prevent group members from disrupting the security of the service.

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 (1), 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 (1), then when he observes a rekeying message of 64 000 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 a related paper, we have identified other system weaknesses that can occur in multicast key distribution schemes even when the underlying cryptographic algorithm is provably correct [17].

B. 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 [18], [19]. Data embedding 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 imperceptibly embed data in the cover signal, allow for easy extraction of the embedded information, and achieve a high embedding rate. Data embedding schemes that have the property that it is impossible for an observer to detect the presence of an information signal in the cover signal are referred to as *steganographic*.

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 [5]–[7]. 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 [6], 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/s in NTSC signals. Another method for embedding information in video was presented in [20], 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 [21], [22]. 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 paper to demonstrate the feasibility of our multimedia multicast key distribution philosophy.

Associated with many embedding schemes is an *embedding key* that 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 [21], [22], 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 [21], [22] 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 nonnegative integer. At random, we shall choose q different embedding rules $(r_0, r_1, \ldots, r_{q-1})$, allowing for repetition in the rules selected. Each embedding rule r_k describes one of the 24 possible ways to map 2 bits to four regions. We assign an embedding rule r_k for each chunk c_j according to $k \equiv j (\text{mod } q)$. Thus, the $0, q, 2q, \ldots$ 2-bit chunks use embedding rule r_0, the $1, q+1, 2q+1, \ldots$ 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 advantages of using data embedding to convey rekeying messages compared to a media-independent channel is that data embedding can maintain the data rate and 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, the presence of an additional channel leads to an increase in the bandwidth needed to provide the multimedia service. An adversary who observes the data as they are transmitted may detect the additional bandwidth to infer that a key updating operation is occurring. As noted earlier, by observing the external channel an adversary can 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 infer information about the group membership. From a security point of view, this provides valuable information to a potential adversary. In order for the media-independent channel to maintain the original data rate, it is necessary to perform computationally expensive transcoding operations.

In comparison, when using a media-dependent channel, the data embedding operation provides a graceful degradation of media quality, and makes it possible to maintain the original data rate of the media without performing the additional computations associated with transcoding. Data embedding easily provides *covert* information transferral when used in conjunction with the encryption of the multimedia data. Encrypting the embedded media stream will also preserve the bit rate, and produce an output that hides the presence of an embedded signal since, for an adversary to be able to detect the presence of embedded content, he must be able to infer information about the plaintext given the ciphertext. This is not possible for reasonable ciphers, such as DES and Rijndael, and therefore it is impossible for an eavesdropper to measure information regarding the occurrence of a rekeying operation. We emphasize that it is not necessary for the data embedding scheme to be steganographic since the data embedding operation occurs prior to the encryption of the multimedia. The primary concerns for choosing an embedding technique for the media-dependent channel are therefore the embeddability rate and the imperceptibility of the embedding.

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 adversary 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 [23] and it might not be possible to increase the length of the session key.

When using media-dependent channels, the issue of reliability becomes more pronounced than in the media-independent 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 network level delivery protocols employing retransmissions to improve the reliability of key delivery. There is therefore a tradeoff between the covert information transferral provided by the media-dependent channel and the reliability provided by the media-independent channel when delivering the rekeying message.

It is possible, however, to address the reliable delivery of the rekeying messages when using the media-dependent channel by employing a retransmission scheme at the application layer. For example, the multimedia system may employ a centralized error recovery technique similar to the NP protocol of [24], 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 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 acknowledgment messages to the server. In order to maintain the covertness advantage of data embedding, the acknowledgment messages should be relayed back to the sender using an anonymous routing technique, such as provided by Onion Routing [25]. 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.

V. 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 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.* [26] and Blundo *et al.* [27] each present a method using polynomial interpolation whereby the broadcast message does not have a partitioned structure like the message in (1). 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 V-A, 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 [18]. Additionally, although our use of one-way functions can be applied to the polynomial interpolation methods of [26], [27], 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 V-B, 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, which are the building blocks of our message form.

Definition 1: 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 [14], [23] 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 nonidentity elements of Z_p^*, the multiplicative subgroup of integers modulo p, it is computationally difficult to determine x given $z = y^x (\mathrm{mod}\, p)$ and y [14], [23]. 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 [14], 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 paper we shall assume the existence of POWFs that map sequences of $2B$ bits into sequences of B bits.

A. Basic Message Form

For the basic message form, we shall use the key distribution scheme depicted in Fig. 1. Suppose that at time $t - 2$ the group consists of n users $u_1, u_2, \ldots, u_n$. 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

$$\alpha_\epsilon(t) = K_\epsilon(t) + \prod_{i=1}^{n-1} f(K_i, \mu(t)). \tag{2}$$

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

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 show that it is computationally hard for a member to compute another member's private KEK. First, consider a weak variant this message form:

$$\alpha_\epsilon(t) = K_\epsilon(t) + \prod_{i=1}^{n-1} K_i. \tag{3}$$

This form of the message requires that $K_\epsilon(t) < \min_i\{K_i\}$. Although this form of the message distributes $K_\epsilon(t)$ to all of the needy members, it is possible for user u_i to acquire the keys K_j of other users since he may calculate

$$A_i = \frac{\alpha_\epsilon(t) - K_\epsilon(t)}{K_i} = \prod_{j \neq i} K_j \tag{4}$$

and then factor A_i to acquire information about the other K_j's. If the keys K_j correspond to symmetric encryption keys, then typically the key length will at most be 150 bits, and factoring A_i will not be difficult [14], [23].

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

$$A_i = \prod_{j \neq i} f(K_j, \mu(t)). \tag{5}$$

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.

We now discuss how this message format reduces the communication overhead compared to a partitioned message format, such as is depicted in (1). Current multicast key management schemes, such as [1]–[3], 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 Fig. 1, with $n+1$ users. When using the partitioned message form of (1), 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

$$\rho = \frac{n\log_2 n}{nB_k + n\log_2 n} \tag{6}$$

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 (2) 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 (2) 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 (2) is more efficient in terms of communication. This occurs when we are providing service to a group with more than 2 users. Therefore, the message format of (2) is more efficient in terms of communication.

B. 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 [1], [3].

A binary tree is shown in Fig. 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 $K_{111}, K_{11}, K_1, K_\epsilon$, and K_s. All of the keys, with the possible exception of the leaf keys, may vary with time to reflect 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 III-A 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 [28], a nonbalanced 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 nonfull 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.* [28] one can see how to make an approach handle member joins and departures for nonbalanced trees.

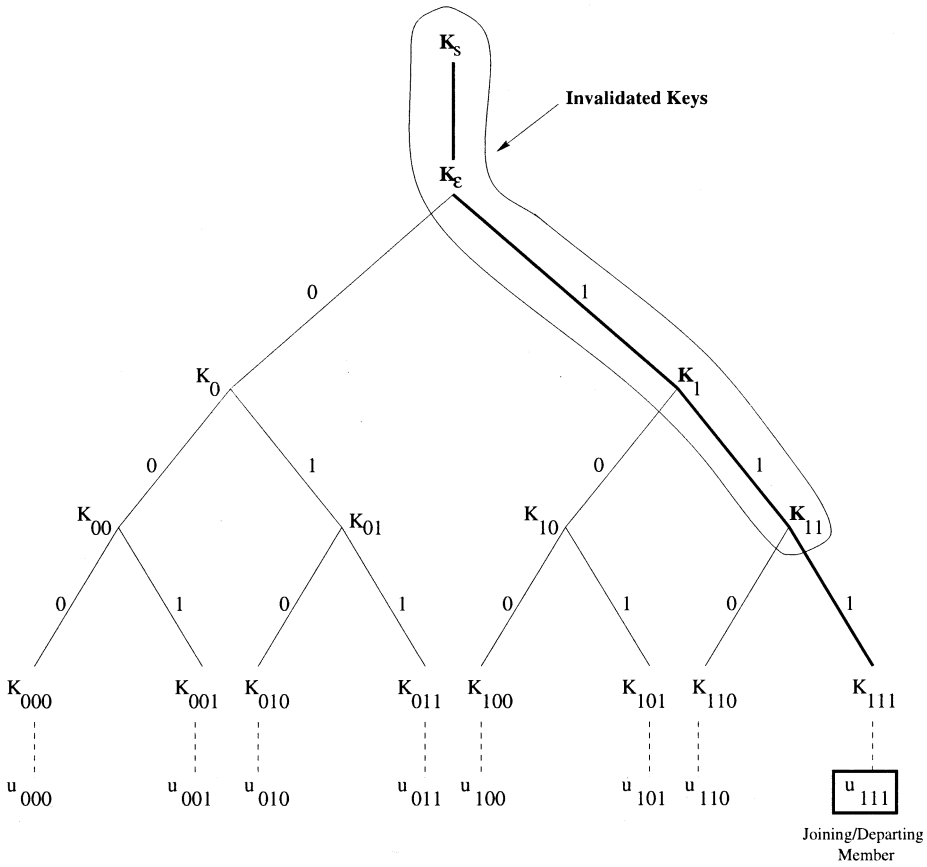


Fig. 5. Tree-based key distribution scheme.

1) Member Join: The member join operation does not involve the message format of (2) since each node of the key tree updates itself. Nonetheless, we present this case for completeness. Consider the binary tree depicted in Fig. 5, that has seven 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 $K_\epsilon(t)$.

2) Member Departure: When a member leaves the group, multiple keys become invalidated because that user shares these keys with other users. For example, in Fig. 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 departure: 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

$$\alpha_\epsilon(t) = K_\epsilon(t) + f(K_0(t-1), \mu(t)) f(K_{10}(t-1), \mu(t)) f(K_{110}, \mu(t)) \quad (7)$$

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

$$\alpha_1(t) = K_1(t) + f(K_{10}(t-1), \mu(t)) f(K_{110}(t-1), \mu(t)) \quad (8)$$

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

$$\alpha_{11}(t) = K_{11}(t) + f(K_{110}(t-1), \mu(t)). \quad (9)$$

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

$$\alpha_{11}(t) = K_{11}(t) + \prod_{j=0}^{0} f(K_{11j}(t-1), \mu(t)). \quad (10)$$

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

$$\alpha_1(t) = K_1(t) + \prod_{j=0}^{1} f(K_{1j}(t), \mu(t)) \quad (11)$$

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

$$\alpha_\epsilon(t) = K_\epsilon(t) + \prod_{j=0}^{1} f(K_j(t), \mu(t)). \quad (12)$$

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

$$\alpha_s(t) = E_{K_\epsilon(t)}(K_s(t)). \quad (13)$$

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

$$\begin{aligned} C_{\mathrm{td}} &= \sum_{i=1}^{L} i(a-1) \\ &= (a-1)\frac{\log_a n(log_a n+1)}{2}. \end{aligned} \quad (14)$$

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

$$\begin{aligned} C_{\mathrm{bu}} &= aL - 1 \\ &= a\log_a n - 1. \end{aligned} \quad (15)$$

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.

VI. 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 ten 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 V. 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 V 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 V-B, the amount of multiplications (computation) needed to update the KEKs for the bottom-up approach was calculated to be $C_{\mathrm{bu}} = a\log_a n - 1$. 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

$$S = \frac{a^{L+1}-1}{a-1} \quad (16)$$

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 V-B via an internal channel. If we denote R as the embeddable channel rate (in bits/s), 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

$$T = \frac{C_{\mathrm{bu}}B_{\mathrm{KEK}} + B_s + B_{\mathrm{emb}} + B_\mu}{R}. \quad (17)$$

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

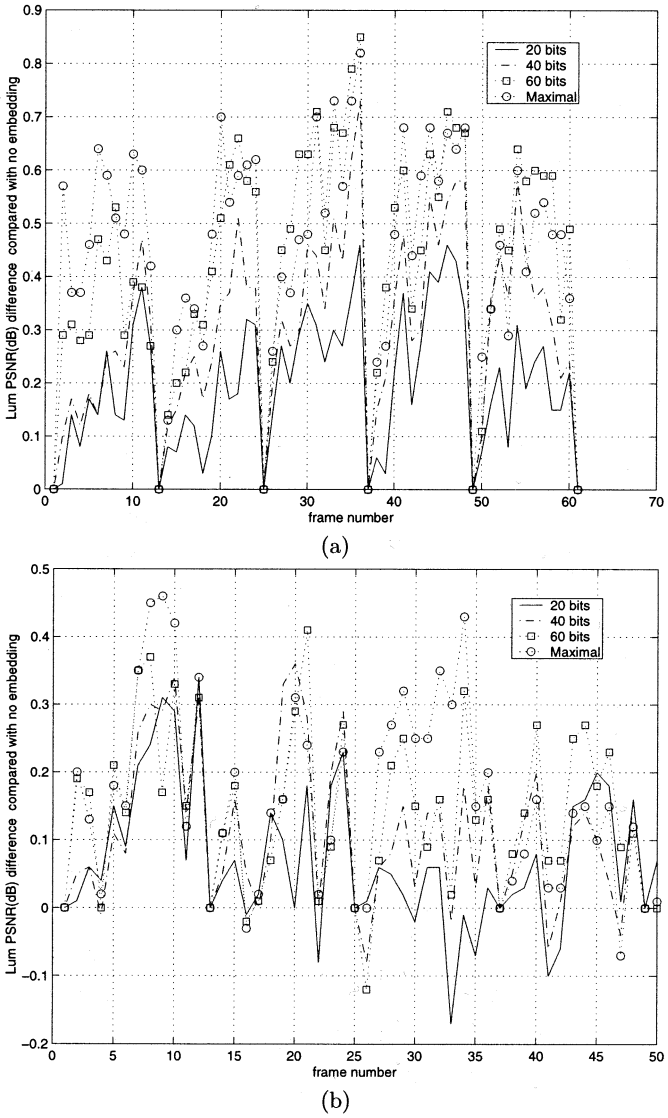


Fig. 6. Peak signal-to-noise ratio (PSNR) difference of the luminance components between no embedding and the embedding scheme of [22] with variable embedding rate. (a) Foreman and (b) Miss America.

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 [22]. 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 [29]. The bitrate in the simulation is 64 kbps with a frame rate 10 f/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 Fig. 6(a) for *Foreman* and Fig. 6(b) for *Miss America*. Their average PSNR differences are also listed in Table I. In all cases,

TABLE I
AVERAGE PSNR DIFFERENCE

	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)

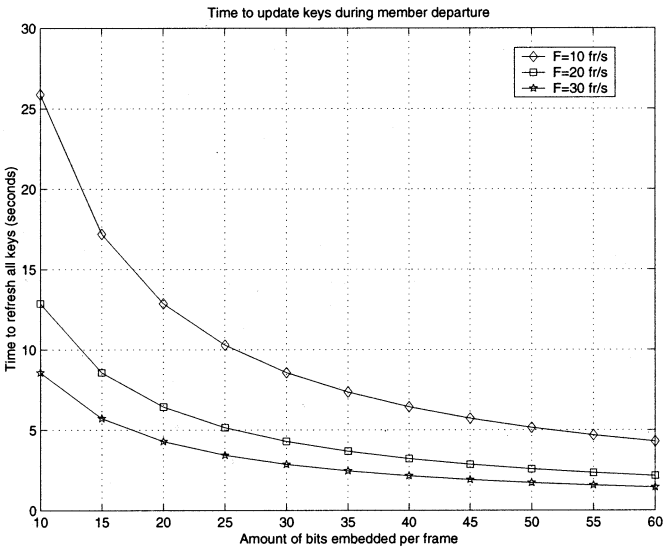


Fig. 7. The time needed to refresh the entire set of keys during a member departure using the bottom-up approach with different frame rates F, and different amounts of bits embedded per frame. The group size is $n = 2^{20}$, or roughly one million users.

the PSNR degradation of Luminance is within 1 dB for both *Foreman* and *Miss America*, which normally cannot be detected by human visual system for video applications. Additionally, it was shown in [22] that data embedding at half-pel motion estimation at most degenerates the video coding 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 $B_{\mathrm{KEK}} = 56$ bits, $B_s = 56$ bits, $B_\mu = 56$ and $B_{\mathrm{emb}} = 20$ 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 block cipher DES. The resulting times needed to refresh the keys are presented in Fig. 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/s, and desire to refresh the keys during member departure in $T = 5$ s, 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 1 dB 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

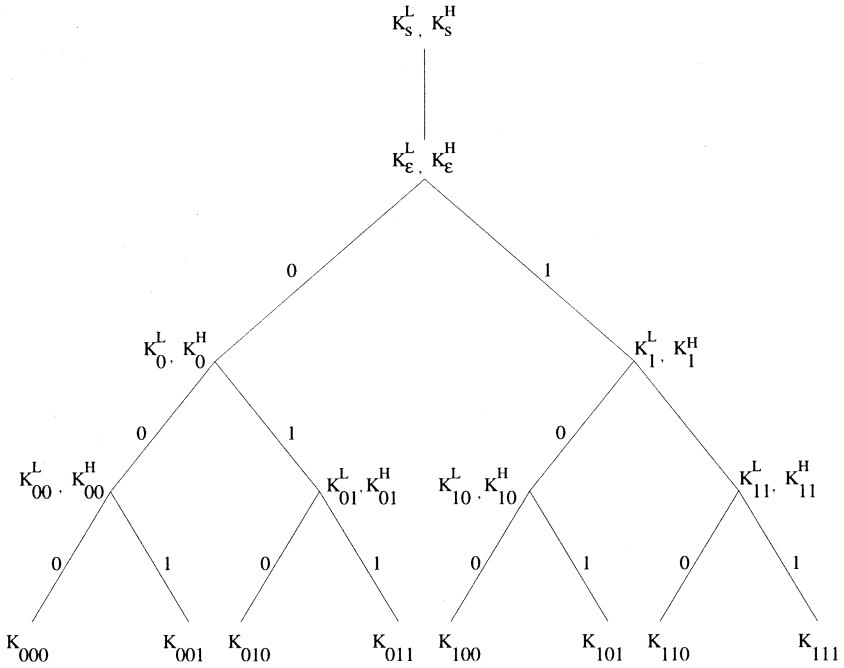


Fig. 8. Key distribution scheme for multilayer multimedia multicast.

rekey larger group sizes, refresh keys faster, or increase the protection by using larger key lengths.

VII. Extensions to Multilayered Services

In many application environments, the multimedia data is distributed in a multilayered 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 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 multilayered, or multiobject 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 V-B and consider the problem of managing keys for two levels of service corresponding to a low quality and high quality service, as depicted in Fig. 8. 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 can be obtained by receiving both the base-layer D^l and the refinement-layer D^h. The GC will have two session keys $K_s^l(t)$ and $K_s^h(t)$. $K_s^l(t)$ is used to encrypt D^l and $K_s^h(t)$ is used to encrypt D^h. Similarly, each internal node in the key tree has two internal keys $K_\sigma^l(t)$ and $K_\sigma^h(t)$, 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 $K_\epsilon^l(t)$ and transmitting the message $\alpha_s^l(t) = E_{K_\epsilon^l(t)}(K_s^l(t))$.
- *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 V-B1.
- *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 V-B1 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 V-B 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 V-B 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 V-B once on the high-layer internal keys.

VIII. Conclusion

The secure distribution of multimedia multicasts necessitates the distribution and management of keying material. In this paper, we have presented two modes of conveyance for transmitting the rekeying messages. Typically, rekeying information 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.

The rekeying messages need to make efficient usage of communication resources, and must be robust to attacks by both nonmembers as well as members. A new form for the rekeying messages that employs one-way functions and a broadcast seed was presented that does not require the typical additional communication overhead needed to identify which portion of a rekeying message is intended for which user. Our proposed format of the rekeying message can withstand collusion and allows for user-specific information to be reused. When the group size becomes large, efficient usage of communication resources is achieved by mapping the message form to a logical tree.

We then used the 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/s, we observed that it was possible to refresh the keys for a group size of roughly one million users in 5 s when we used an embeddability rate of 25 bits/frame. The distortion introduced to the video sequence was less than 0.8 dB 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.

References

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

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

[3] 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.

[4] R. Canetti, T. Malkin, and K. Nissim, "Efficient communication-storage tradeoffs for multicast encryption," *Eurocrypt*, pp. 456–470, 1999.

[5] I. Cox, J. Kilian, F. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Tran. Image Processing*, vol. 6, pp. 1673–1687, Dec. 1997.

[6] F. Hartung and B. Girod, "Digital watermarking of MPEG-2 coded video in the bitstream domain," in *IEEE ICASSP'97*, 1997, pp. 2621–2624.

[7] C. Podilchuk and W. Zeng, "Image adaptive watermarking using visual models," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 525–540, May 1998.

[8] M. Wu and B. Liu, "Modulation and multiplexing techniques for multimedia data hiding," in *Proc. SPIE ITcom'01*, vol. 4518, Aug. 2001.

[9] A. Puri and T. Chen, *Multimedia Systems, Standards, and Networks*. New York: Marcel Dekker, 2000.

[10] S. Paul, *Multicasting on the Internet and Its Applications*. Norwell, MA: Kluwer, 1998.

[11] R. Poovendran and J. S. Baras, "An information theoretic approach for design and analysis of rooted tree-based multicast key management schemes," in *Advances in Cryptology: Crypto'99*, 1999, pp. 624–638.

[12] U. Horn, K. Stuhlmller, M. Link, and B. Girod, "Robust internet video transmission based on scalable coding and unequal error protection," *Image Commun.*, vol. 15, pp. 77–94, Sept. 1999.

[13] H. Zheng and K. J. R. Liu, "Optimization approaches for delivering multimedia services over digital subscriber lines," *IEEE Signal Processing Mag.*, vol. 17, pp. 44–60, July 2000.

[14] W. Trappe and L. C. Washington, *Introduction to Cryptography with Coding Theory*. Upper Saddle River, NJ: Prentice-Hall, 2002.

[15] 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. New York: Marcel Dekker, 2000, pp. 367–405.

[16] J. Mitchell, W. Pennebaker, C. Fogg, and D. LeGall, *MPEG Video Compression Standard*. London, U.K.: Chapman & Hall, 1997.

[17] R. Poovendran and J. S. Baras, "An information-theoretic approach for design and analysis of rooted-tree-based multicast key management schemes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2824–2834, 2001.

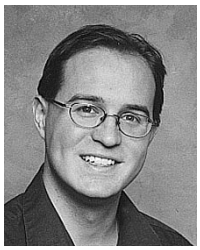
[18] W. Trappe, J. Song, R. Poovendran, and K. J. R. Liu, "Key distribution for secure multimedia multicasts via data embedding," presented at the IEEE ICASSP, 2001.

[19] J. Song, R. Poovendran, W. Trappe, and K. J. R. Liu, "A dynamic key distribution scheme using data embedding for secure multimedia multicast," presented at the SPIE 2001, Security and Watermarking for Multimedia, San Jose, CA, 2001.

[20] A. Westfeld and A. Pfitzmann, "Attacks on steganographic systems," presented at the 2nd Int. Workshop on Information Hiding, 1998.

[21] J. Song and K. J. R. Liu, "A data embedding scheme for H.263 compatible video coding," in *Proc. IEEE ISCAS*, vol. 4, June 1999, pp. 390–393.

[22] ——, "A data embedded video coding scheme for error-prone channels," *IEEE Trans. Multimedia*, vol. 3, pp. 415–423, Dec. 2001.
[23] *Handbook of Applied Cryptography*, CRC, Boca Raton, FL, 1997.
[24] 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.
[25] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous connections and onion routing," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 482–494, May 1998.
[26] M. Just, E. Kranakis, D. Krizanc, and P. vanOorschot, "On key distribution via true broadcasting," in *Proc. 2nd ACM Conf. Computer and Communications Security*, 1994, pp. 81–88.
[27] C. Blundo, L. A. Frota Mattos, and D. R. Stinson, "Multiple key distribution maintaining user anonymity via broadcast channels," *J. Comput. Security*, vol. 3, pp. 309–323, 1994.
[28] D. Balenson, D. McGrew, and A. Sherman, "Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization,", Internet Draft Report.
[29] *Video Coding for Low Bitrate Communication, Ver. 2*, Jan. 1998.



Wade Trappe (M'02) received the B.A. degree in mathematics from The University of Texas at Austin in 1994 and the Ph.D. in applied mathematics and scientific computing from the University of Maryland, College Park, in 2002.

He is currently an Assistant Professor at the Wireless Information Network Laboratory (WINLAB) and the Electrical and Computer Engineering Department, Rutgers University, Piscataway, NJ. His research interests include multimedia security, information and network security, and computer networking. He is a co-author of the textbook *Introduction to Cryptography with Coding Theory* (Upper Saddle River, NJ: Prentice-Hall, 2001).

Dr. Trappe received the George Harhalakis Outstanding Systems Engineering Graduate Student award while at the University of Maryland. He is a member of the IEEE Signal Processing and Information Theory societies, and a member of the Society for Industrial and Applied Mathematics.

Jie Song was born in Taiyuan, China, on October 31, 1968. He received the B.S. degree from Beijing University, Beijing, China in 1990 and the M.S. degree from Beijing University of Posts and Telecommunications (BUPT) in 1993, and the Ph.D. degree from University of Maryland, College Park, in 2000, all in electrical engineering.

From April 1993 to June 1996, he was a Lecturer and Researcher in the Information Engineering Department at BUPT. He worked for Fujitsu Labs of America, Sunnyvale, CA, in the summer of 1997. From November 1997 to February 1999, he was a part-time consultant of multimedia technologies with Odyssey Technologies, Inc., Jessup, MD, where he involved in the projects of H.323/H.324 videophone, portable multimedia terminal design and multichannel video capturing systems. Since August 2000, he has been working on research, design and implementation for broadband and satellite communication systems at Agere Systems (formerly microelectronics group, Lucent Technologies), Holmdel, NJ. His research interests include signal processing for digital communication and multimedia communications.

Radha Poovendran (M'00) received the Ph.D. in electrical engineering from the University of Maryland, College Park (UMCP), in 1999.

After spending a year as a Research Staff Member at UMCP, he joined the Department of Electrical Engineering, at University of Washington, Seattle, in September 2000 as an Assistant Professor. His research interests are in all aspects of cryptography and network security with emphasis on multicast and wireless security.

Dr. Poovendran received the Lucite Rising Star Award from the National Security Agency in 1999, the Graduate Student Service Award from the ECE Department of UMCP in 1999, the Faculty Early CAREER Award from the CISE/ANIR division of the National Science Foundation in 2001, the U.S. Army Research Office Young Investigator Award in 2002, and both the Outstanding Teaching Award and the Outstanding Research Advisor Award from the EE Department of the University of Washington in 2002. In 2003, he received the Outstanding Faculty Award from the College of Engineering at the University of Washington.

K. J. Ray Liu (F'03) received the B.S. degree from the National Taiwan University, Taipei, Taiwan, R.O.C., in 1983, and the Ph.D. degree from the University of California, Los Angeles, in 1990, both in electrical engineering.

He is Professor with the Electrical and Computer Engineering Department and the Institute for Systems Research, University of Maryland, College Park. His research interests span broad aspects of signal processing algorithms and architectures; multimedia communications and signal processing; wireless communications and networking; information security; and bioinformatics, in which he has published over 280 refereed papers. He was the founding Editor-in-Chief of *EURASIP Journal on Applied Signal Processing* and an editor of *Journal of VLSI Signal Processing Systems*.

Dr. Liu is the Editor-in-Chief of IEEE SIGNAL PROCESSING MAGAZINE and has been an Associate Editor of IEEE TRANSACTIONS ON SIGNAL PROCESSING, a Guest Editor of special issues on Multimedia Signal Processing of PROCEEDINGS OF THE IEEE, a Guest Editor of special issue on Signal Processing for Wireless Communications of IEEE JOURNAL OF SELECTED AREAS IN COMMUNICATIONS, a Guest Editor of special issue on Multimedia Communications over Networks of IEEE SIGNAL PROCESSING MAGAZINE, and a Guest Editor of special issue on Multimedia over IP of IEEE TRANSACTIONS ON MULTIMEDIA. He has served as Chairman of the Multimedia Signal Processing Technical Committee of the IEEE Signal Processing Society. He is the recipient of numerous honors and awards, including the IEEE Signal Processing Society 2004 Distinguished Lecturer, the 1994 National Science Foundation Young InvestigatorAward, the IEEE Signal Processing Society's 1993 Senior Award (Best Paper Award), IEEE 50th Vehicular Technology Conference Best Paper Award, Amsterdam, 1999. He also received the George Corcoran Award in 1994 for outstanding contributions to electrical engineering education and the Outstanding Systems Engineering Faculty Award in 1996 in recognition of outstanding contributions in interdisciplinary research, both from UMD.