KEY DISTRIBUTION FOR SECURE MULTIMEDIA MULTICASTS VIA DATA EMBEDDING

Wade Trappe, Jie Song ,Radha Poovendran†,K.J.Ray Liu*

Department of Electrical and Computer Engineering & Institute for Systems Research
University of Maryland
College Park, MD 20742

ABSTRACT

The problem of controlling access to multimedia multicasts requires the distribution and maintenance of keying information. The conventional approach to distributing keys is to use a channel independent of the multimedia content. We propose a second approach that involves the use of an data-dependent channel, and can be achieved for multimedia by using data embedding techniques. Using data embedding to convey rekeying messages can provide an additional layer of security when compared with the traditional approach. We then introduce multicast key distribution, and employ a recent tree-based key distribution scheme to exhibit the factors involved in transmitting keys using data embedding.

1. INTRODUCTION

Well-developed multimedia standards have created many new applications, and are allowing users to share content and express their creativity. Many future multimedia technologies will involve group-based scenarios, where users work and play together. The most relevant enabling network technology for group communication is multicast. Multicast communications is efficient, reducing 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 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 use of this data by non-privileged users.

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. In order to secure multicast communications, all the members of the group share a common session encryption key (SK). When changes in the group membership occur,the SK must be changed. In order to update the SK, a party responsible for distributing the keys, called the group center (GC), must securely communicate the new key material to the valid users. This task is achieved by transmitting rekeying messages that use key encrypting keys (KEKs) to encrypt and distribute new keys. In addition, any solution to access control should address issues of resource scalability for scenarios consisting of large privileged groups.

2. KEY DISTRIBUTION FOR MULTIMEDIA

There are two types of channels available for distributing the key information needed to secure multimedia multicasts, which are depicted in Figure 1. The first approach is to use a *media-independent channel.* By this we mean that a separate channel needs to be used to convey the keying material. As an example, the specification of the MPEG-4 bitstream allows for the distribution of keying information via Intellectual Property Management and Protection Descriptors (IPMP-Ds) and Intellectual Property Management and Protection Elementary Streams (IPMP-ESs)[1].

We propose a *media-dependent* approach to transmitting the rekeying information that is accomplished using steganographic methods. In these cases, the rekeying information may be embedded in the content and distributed to those who receive the data.

Data embedding, or digital steganography, techniques allow for an information signal to be *hidden* in content without dramatically distorting the content. Effective data embedding techniques are those that can *invisibly* embed data, allow for easy extraction, 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 [2] [3] [4].

Associated with many embedding schemes is an embedding key that governs how the information is embedded into the cover signal. For example, in [5], 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 ways to do this. We may therefore associate an 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.

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, also must be updated. By using data embedding to convey the rekeying messages, an additional layer of security is available to the system. When data embedding is used, an *external* adversary will not only have to attack the SK and KEKs, but he will also have to attack

*Currently at: Lucent Technologies, Room 1K446, 101 Crawfords Corner Rd., Holmdel, NJ 07733

†Currently at: Department of Electrical Engineering,University of Washington, Seattle, WA 98195

0-7803-7041-4/01/$10.00 ©2001 IEEE

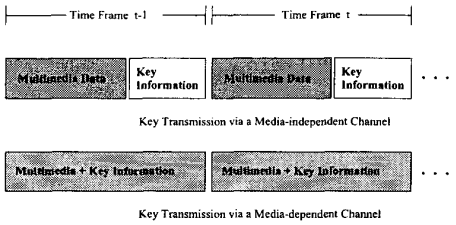

Fig. 1. Two approaches to distributing the key information in multimedia multicasting.

the key governing the embedding rule in order to acquire rekeying messages. It is thus 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.

3. MULTICAST KEY MANAGEMENT

In this Section we describe a basic key management scheme that can be used to update and maintain keys used for securing multicast. The scheme is used to describe the operations needed to transmit rekeying messages by data embedding. The multicast key message form described in this section is detailed in [6], where it is referred to as the Residue-based method. We refer the reader to this paper for more detail.

Consider the single sender multiple receiver multicast model, where a group of n multimedia users will share a multimedia multicast. In the simple key distribution scheme for$n$ sers, depicted in Figure 2, user u_j has two key encrypting keys K_j and K_ϵ, and the session key K_s. The session key K_s is used to encrypt bulk quantities of multimedia content. 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. Both KEKs and the SK are assumed to be B bits in length.

We first define parametric one-way functions.

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[7] that are parameterized by the parameter y. The discrete logarithm provides an example of a POWF [7]. Throughout this paper we shall assume the existence of parametric one-way functions that map sequences of $2B$ bits into sequences of B bits.

The group center makes available a POWF h that maps a sequence of $2B$ bits to B bits. Define a new function f that prepends 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 in equation (1) will yield $K_\epsilon(t)$.

There are three types of key update scenarios: key refreshing, member joins, and member departure. These scenarios arise due to the expiration of security lifetime of keying material, and the invalidation of shared key material due to membership dynamics. We shall only describe the procedure for updating keys using data embedding, which is depicted in Figure 3. The procedure for updating keys using a media-independent channel is a straight forward modification of the media-dependent procedure.

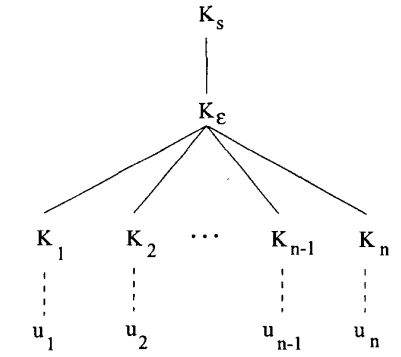

Fig. 2. The arrangement of keys used in the basic form of the residue-based method.

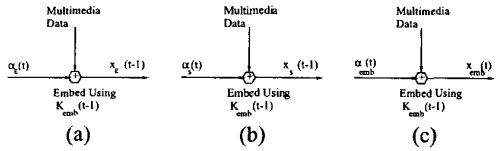

Fig. 3. The steps involved in updating the key information during member changes using data embedding. Step (a) Updating the root KEK, (b) Updating the session key, and (c) Changing the embedding rule.

Key Refresh: In order to limit the risk of key compromise, it is necessary to refresh keys prior to key expiration. The refreshing operations will take place during time interval $t - 1$. In the key refreshing stage, it is not necessary to renew the KEKs, thus $K_\epsilon(t) = K_\epsilon(t-1)$. 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 have used the notation $E_K(m)$ to denote the encryption of message m with key K. The message $\alpha_s(t)$ is embedded in the multimedia data to produce a composite signal $x_s(t)$, which is encrypted using the old session key $K_s(t-1)$ and is then broadcast. Knowledge of $K_{emb}(t-1)$, $K_s(t-1)$ and $K_\epsilon(t)$ is needed to acquire $K_s(t)$.

Member Join: 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. We must renew both the SK and the root KEK in order to prevent the new user from accessing previous communication. The first stage 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 the message $\alpha_\epsilon(t) = E_{K_\epsilon(t-1)}(K_\epsilon(t))$. The message $\alpha_\epsilon(t)$ is embedded in the content, the composite data is encrypted and broadcast to all users. Next, the session key is updated to $K_s(t)$. Since all of the current members have $K_\epsilon(t)$, the session rekeying message $\alpha_s(t) = E_{K_\epsilon(t)}(K_s(t))$ is embedded in the content, the composite signal is encrypted, and broadcast.

Member Departure: Suppose, without loss of generality, that user n decides to leave at time $t - 2$, then both $K_\epsilon(t-1)$ and

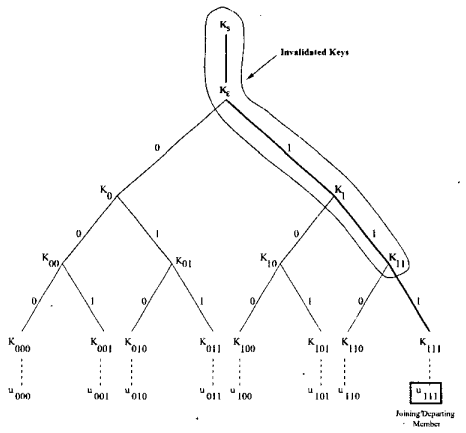

Fig. 4. Tree-based key distribution.

$K_s(t-1)$ 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 an B-bit random seed $\mu(t)$ using any available channel. Next, the GC forms $K_\epsilon(t)$ and calculates the rekeying message as

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

which is embedded in the content. The composite data is encrypted and transmitted. A legitimate member u_i may decode $\alpha_\epsilon(t)$ to get $K_\epsilon(t)$ by decrypting the composite signal using $K_s(t-1)$, extracting message using the embedding key $K_{emb}(t-1)$, and calculating $\alpha_\epsilon(t) \pmod{f(K_i, \mu(t))}$. The session key is then updated by embedding $\alpha_s(t) = E_{K_\epsilon(t)}(K_s(t))$, encrypting the composite signal, and broadcasting.

In any of these operations, the GC may update the embedding key by embedding $\alpha_{emb}(t) = K_{emb}(t)$ in the multimedia stream by using the previous embedding rule. The composite signal $x_{emb}(t)$ is encrypted using $K_s(t-1)$ and broadcast to current group members.

4. SCALABILITY

When the multicast group is very large, the previous methods have severe computational and communication overhead associated with member departures because the size of α_ϵ increases linearly with the number of users n. One popular approach to improving scalability is to distribute the keys according to a tree structure.

A binary tree is shown in Figure 4, 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 which is indexed by the path leading to itself. The symbol ϵ is used to denote the root node. Each user is assigned to a leaf and is given the KEKs of the nodes from the leaf to the root node in addition to the session key. For example, user u_{111} is assigned keys $K_{111}, K_{11}, K_1, K_\epsilon$,an d K_s.

The residue-based method can be modified to achieve desirable scalability properties during member changes. We exhibit this for the member departure case. When a member leaves the group, multiple keys become invalidated because that user shares these keys with other users. 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; other keys which aren't invalidated are reused for the next time frame. Additionally, K_{111} does not need to be updated since it not shared with any other users.

The most efficient approach to updating the keys is to update them from the leaf node to the root node (bottom-up). The keys are updated in the order K_{11}, K_1,an d K_ϵ. After updating the key encrypting keys, the root KEK $K_\epsilon(t)$ can be used to encrypt the new session key $K_s(t)$.

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

$$\alpha_{11}(t) = K_{11}(t) + f(K_{110}(t-1), \mu(t)), \tag{2}$$

which is embedded in the content. The next key that is updated is $K_1(t-1)$. Since two of the users beneath K_1 share a common key $K_{10}(t) = K_{10}(t-1)$ that is not invalidated by the departure of member u_{111}. For efficiency, we use this key in updating K_1. The resulting message

$$\alpha_1(t) = K_1(t) + \prod_{j=0}^{1} f(K_{1j}(t), \mu(t)) \tag{3}$$

is embedded in the content. 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 embed the message

$$\alpha_\epsilon(t) = K_\epsilon(t) + \prod_{j=0}^{1} f(K_j(t), \mu(t)). \tag{4}$$

Finally, the session key is updated by encrypting the new session key $K_s(t)$ using the new root KEK $K_\epsilon(t)$, and embedding the message $\alpha_s(t) = E_{K_\epsilon(t)}(K_s(t))$ in the content. The total amount of communication needed to update the entire system of keys when using an a-ary tree is $(B+1)(a \log_a n - 1)$ which is a considerable savings compared to the methods of Section 3 as n becomes large.

5. 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.

A tree-based key distribution scheme has good communication scalability properties. 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. 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 might be appropriate. However, although a tree-based scheme may require more storage for each user,an d 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 4, the amount of multiplications 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

$$S = \frac{a^{L+1} - 1}{a - 1} \tag{5}$$

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

Using either type of 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 4 via an media-dependent 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

$$T = \frac{C_{bu} B_{KEK} + B_s + B_{emb} + B_\mu}{R}. \tag{6}$$

Since T is related to the bit size of each of the keys, it is therefore related to the security levels protecting the service. The time T 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, the time needed to update keys is also critical.

Using a generic data embedding scheme in conjunction with the member departure scheme of Section 4, we calculated the amount of time needed to refresh the entire network of keys for a tree of degree $a = 2$,an d $n = 2^{20}$ 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 to correspond to the DES key size. The times needed to refresh the keys are presented in Figure 5. 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 with a frame rate of $F = 20$ frames/second, and desire to refresh the keys during departures in $T = 5$ seconds, then 25 bits must be embedded per frame.

6. CONCLUSIONS

The secure distribution of multimedia multicasts necessitates the distribution and management of keying material. The rekeying messages may be distributed by means of either a media-independent channel or through a media-dependent channel using data embedding. The proposed use of data embedding for transmitting keying

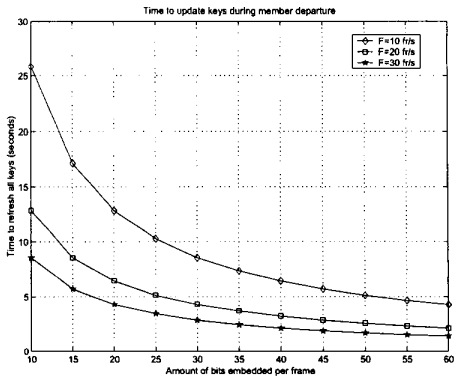

Fig. 5. The time needed to refresh the entire set of keys during a member departure using the bottom-up approach with different frame rates F,an d different amounts of bits embedded per frame. The group size is $n = 2^{20}$, or roughly one-million users.

information provides an additional layer of security: adversaries must successfully attack the key governing the data embedding rule prior to being able to access and attack the rekeying messages.

A key distribution scheme, as well a scalable analog, were presented and used to illustrate the operations needed to update the keys when using data embedding. The relationship between the amount of time needed to update the keying network and the bitsize of the various keys was established for the key distribution schemes presented. The embeddability rate plays an important role in governing what security levels can be achieved since lower embeddability rates necessarily means that the key lengths must be shorter in order to refresh the key network in the same amount of time.

7. REFERENCES

[1] 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.Ch en, Eds., pp. 367–405. Marcel Dekker Inc., 2000.

[2] 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.

[3] 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.

[4] F. Hartung and B. Girod, "Digital watermarking of MPEG-2 coded video in the bitstream domain," *IEEE Int. Conf.Acco stic Speech and Signal Proc. '97*,p p. 2621–2624, 1997.

[5] 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.

[6] W. Trappe, R. Poovendran, J. Song, and K.J.R. Liu, "Reusable multicast key distribution schemes for dynamic groups," in *IEEE Int. Symp. on Inf. Theory*,2 001, Submitted.

[7] A. Menezes, P. vanOorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.