CASM: A Content-Aware Protocol for Secure Video Multicast

Hao Yin, Chuang Lin, Feng Qiu, Jiangchuan Liu, Geyong Min, Bo Li

***Abstract*—Information security has been a critical issue in the design and development of reliable distributed communication systems and has attracted significant research efforts. A challenging task is how to maintain information security at a high level for multiple-destination video applications with the huge volume of data and dynamic property of clients. This paper proposes a novel Content-Aware Secure Multicast (CASM) protocol for video distribution that seamlessly integrates three important modules: 1) a scalable light-weight algorithm for group key management; 2) a content-aware key embedding algorithm that can make video quality distortion imperceptible and is reliable for clients to detect embedded keys; and 3) a smart two-level video encryption algorithm that can selectively encrypt a small set of video data only, and yet ensure the video as well as the embedded keys unrecognizable without a genuine key. The implementation of the CASM protocol is independent of the underlying multicast mechanism and is fully compatible with existing coding standards. Performance evaluation studies built upon a CASM prototype have demonstrated that CASM is highly robust and scalable in dynamic multicast environments. Moreover, it ensures secure distribution of key and video data with minimized communication and computation overheads. The proposed content-aware key embedding and encryption algorithms are fast enough to support real-time video multicasting.**

***Index Terms*—Secure multicast, Selective encryption, Key management, Content-awareness**

I. Introduction

Due to the multiple-destination nature of video programs, real-time video multicast has received great interests and significant research efforts over the past decade [1]. Information security has been a critical issue in the design and development of distributed systems and become more crucial for open and dynamic group communications in the IP multicast environment [3, 4, 14, 17]. For overlay or application-layer multicast, the security problem persists if trust relation is not established among the participating nodes. However, conventional methods for secure communications, like direct data encryption, often fail due to the large group and video data sizes, as well as the dynamic property of clients.

In this paper, we propose a novel Content-Aware Secure Multicast (CASM) protocol for video distribution. CASM seamlessly integrates three important modules: 1) a scalable light-weight algorithm for group key management; 2) a content-aware key embedding algorithm that can yield imperceptible video quality distortion and is reliable for clients to detect embedded keys; and 3) a smart two-level video encryption algorithm. This modularized design enables asynchronous updates of individual components in the CASM system. Furthermore, the implementation of the CASM protocol is independent of the underlying multicast mechanism and is fully compatible with existing coding standards. We have implemented a prototype of CASM demonstrating that this protocol is highly robust and scalable in dynamic multicast environments. Moreover, it ensures secure distribution of key and video data with minimized communication and computation overheads.

The rest of this paper is organized as follows. Section II presents the background and related work. An overview of CASM is outlined in Section III. The new key management, embedding, and secure transmission modules are presented in Sections IV, V, and VI, respectively. Section VII discusses the implementation of CASM and presents experimental results. Finally, Section VIII concludes the paper.

II. Background And Related Work

This section will present some preliminaries of secure video multicast and report existing research work.

A. Group Key Management

The goal of key management is to securely share and update the keys for data encryption/decryption among a group of clients. A comprehensive survey on the schemes of group key management can be found in [12]. Among the existing approaches, Efficient Large-Group (ELK) algorithm [13] using a balanced binary key tree to manage the keys is particularly interesting. The distinguishing feature of ELK is that its rekey messages are of a fixed and short length that can thus be easily embedded into video frames. ELK ensures reliable rekey message distribution and provides hints for key recovery, but it suffers from excessive overheads. The CASM proposed by this study achieves reliable message distribution using the other two modules; hence, the overhead of its key management module is relatively light-weighted as compared to ELK.

B. Data Embedding

CASM embeds the key updating messages into video data, thus eliminating the need of a separate reliable control channel that is extremely difficult and costly to establish and manage in a distributed multicast environment.

Digital watermarking is a widely studied data embedding

technique [7-9]. However, many of the algorithms can tolerate certain discrepancies after the process of decoding. As a result, they do not match the demands of the key embedding module of CASM. For consistent data embedding, Alattar et al. [5] have suggested that the original compressed stream can be partially decoded to expose its syntactic elements; such data as Discrete Cosine Transform (DCT) coefficients can then be modified to insert the watermark. The key embedding algorithm in CASM was motivated by such proposals. It modulates the average luminance value of certain regions in I-frames to embed data. We also make effective use of the embedded data for channel coding, which, together with the key management module, achieve highly efficient and reliable key updating for multicast groups.

C. *Selective Encryption*

The high data rate and long playback duration of a video stream yield a huge volume of data. It is impractical to encrypt the entire stream, thus selective encryption is advocated [15]. The selective encryption algorithm in CASM is closely related to that proposed by Shi and Bhargave [21, 22]. They both employed a secret key to randomly change the sign bits of certain data. The difference is that the algorithm in CASM uses a one-way function to generate the keys according to both the frame number and the session key, which is stronger for resisting statistical attacks. This algorithm is faster than those based on Data Encryption Standard (DES) / International Data Encryption Algorithm (IDEA) [19, 20] and suffers less data expansion [18]. Both advantages of this algorithm are desirable for real-time processing and communications. Furthermore, it is fully compatible with the existing video coding standard and hence does not rely on a customized decoder, as in [16].

III. Overview Of CASM

CASM consists of three major modules, namely, key management, key embedding, and secure transmission. Their hierarchy is illustrated in Fig. 1. This section will outline their functionalities; detailed implementations will be addressed in the subsequent three sections.

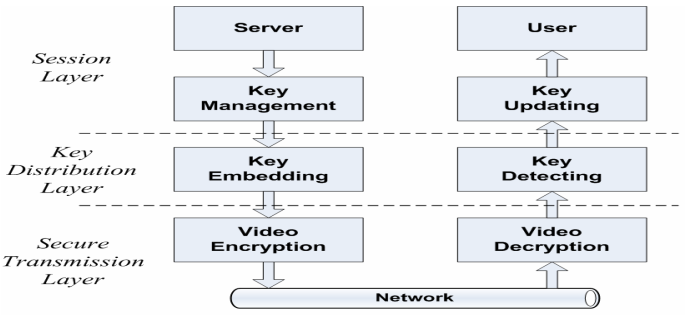


Fig. 1. System hierarchical diagram of CASM.

The key management module manages a family of keys, which form a key tree rooted at the server. Each client corresponds to a leaf node in the tree and records the set of keys along its path from the root. When such events as client joining or leaving occur, the server will update the corresponding keys in the key tree and inform the affected clients to update their key set. As shown in the next section, such organization enables both forward and backward secrecies.

Instead of maintaining a separate control channel, such rekey messages are embedded in the video stream to be distributed to the clients. On the client side, the embedded messages are then detected and forwarded to its key management module.

CASM transmits the video data through the transmission module that ensures secure key and video distribution. Specifically, the video data are encrypted by a selective encryption algorithm with a session key, i.e., the key at the root of the key tree. Since the session key is available at each destination client, it can be used to decrypt the stream, and, if needed, detect the embedded rekey messages. The selective encryption algorithm applies to a small set of video data only, and yet the reconstructed video and the embedded key are generally unrecognizable when decoded with a non-genuine key.

To accommodate current clients joining and leaving the multicast group, CASM divides the running time into slices of identical lengths. The interval for a client to join or leave consists of two slices. In the first, the client to join or leave the system contacts the server for authenticating, and, meanwhile, the server prepares for the new rekey messages. If there are lots of clients joining or leaving simultaneously, key server can use a batch process to update the key tree. In the second slice, the server initiates the key updating process for all the clients joining or leaving in the previous slice, and yet it can accept joining or leaving requests from another batch of clients (as shown in Fig. 2). This pipelining process greatly reduces the frequency of key updating, and consequently, minimizes the communication overhead. The batch process can also greatly reduce the overhead for re-configuration of the key tree. The batch joining process and leaving processing will be discussed respectively in the following sections.

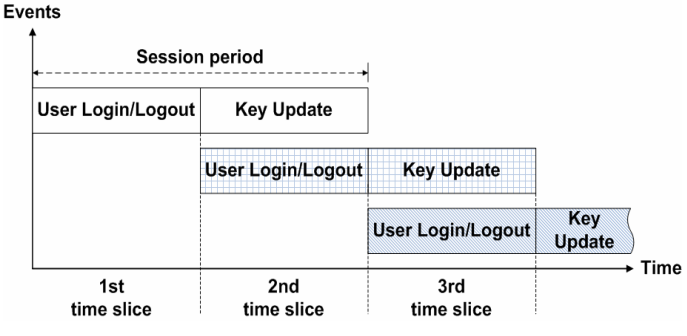


Fig. 2. Illustration of time slices.

IV. THE KEY MANAGMETN MODULE

The key management module of CASM extends the Logical Key Hierarchy (LKH) and One-way Function Tree (OFT) algorithms to achieve secure and efficient key distribution [4]. Similar to Efficient Large-Group (ELK) protocol proposed in [13], CASM maintains a balanced binary key tree. The root of the tree is assigned with the session key for encrypting the video data. Each leaf node corresponds to a legitimate client and is assigned with a key for encrypting the session key and other related keys. A leaf node also records the keys along its path from the root. Fig. 3 illustrates an example of such a tree where client keeps the key set $\{K_G, K_1, K_2, K_4\}$, keeps $\{K_G, K_1, K_2, K_5\}$, and so forth. CASM seamlessly integrates three modules and relies on the key embedding and secure transmission modules to ensure reliable key distribution; its key management module is thus light-weighted as compared to ELK.

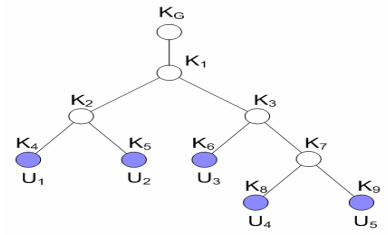


Fig. 3. An example of the key tree.

To facilitate our discussion on key management as well as selective encryption in the next section, we define a family of pseudo-random functions (PRFs) as follows:

$$\mathrm{PRF}^{(m\to n)} : K \times \{0,1\}^m \to \{0,1\}^n$$

Where K is the key, m and n are the lengths of the input and output data, respectively. We also define $\{M\}_K$ as the encrypted sequence of M with key K.

A. Operations for Client Joining

The operations for client joining involve key update for both existing and newly joined nodes. To this end, the following procedure is executed in CASM (see Fig. 4):

1) The new client contacts the server;

2) The server multicasts a message, informing all the present clients to update their keys along their key-tree paths, as follows,

$$K_i' = PRF_{K_i^{\delta}}^{(n\to n)}(K_G), \quad (1)$$

$$K_G' = PRF_{K_G^{\delta}}^{(n\to n)}(0), \quad (2)$$

where $K_i^{\delta} = PRF_{K_i}^{(n\to n)}(\delta)$. This operation ensures backward secrecy, such that the newly joined client cannot decode previous video data;

3) The server generates node N_M for the new client, which is associated with a randomly generated key K_M. Node N_M is then inserted to the key tree. It is either attached to a node with only one child so far, or, if there is no such a node, the server picks leaf node N_j that has the least depth in the key tree, and generates a new parent node N_P for both N_M and N_j. The key K_P for node N_P is set as $PRF_{K_j}^{(n\to n)}$ in Eq. (1), and the server then informs N_j to update its key set;

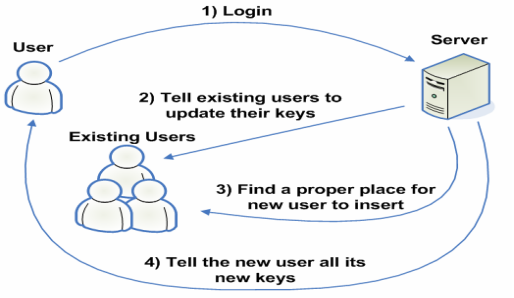


Fig. 4. Procedure for client join.

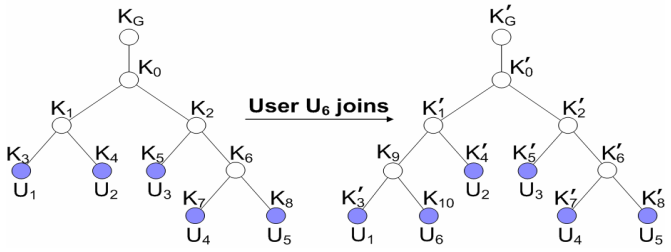


Fig. 5. Illustration of client join.

Fig. 5 shows an example for client U_6 to join the system. Upon receiving the join request from this new client, the server first updates all the present keys in the tree to preserve backward secrecy. Since there is no any node with a single child, the server picks node U_1 and generates a new node with key K_9, which serves as the parent for both U_1 and U_6. The server then informs U_1 to update its key set $\{K_G', K_0', K_1', K_9\}_{K_{10}}$ and sends to U_6.

For joining multiple clients during the same time slice, the key server generates a new sub-tree for these new clients, and attaches this sub-tree to the key tree by the end of this time slice as a normal client does. In this case, the server only needs to communicate with a single node, i.e., the root of the sub-tree, which greatly reduces the communication and computation overhead.

B. Operations for Client Leaving

Upon a client leaving, the current session key has to be replaced and only the legitimate clients can receive the new key. Moreover, to ensure forward secrecy, all the keys that the leaving client knows must also be changed.

The left child contribution: $C_L = PRF_{K_L^\alpha}^{<n \to n_1>}(K)$

The right child contribution: $C_R = PRF_{K_R^\alpha}^{<n \to n_2>}(K)$

$C_{LR} = C_L \mid C_R = PRF_{K_L^\alpha}^{<n \to n_1>}(K) \mid PRF_{K_R^\alpha}^{<n \to n_2>}(K)$

Output: updated key $K' = PRF_{C_{LR}}^{<n \to n>}(K)$

Fig. 6. Algorithm for key update.

We first present a key update algorithm that facilitates the operations for client leaving (Fig. 6). This algorithm is based on the assumption that the two children of the node with key K have keys K_L and K_R, respectively. The new key K' is derived from K together with n_1-bit contribution from K_L and n_2-bit from K_R, $(n_2 + n_2 \le n)$. The server then multicasts the following rekey messages through the secure transmission module:

$$\{C_L\}_{K_R^\beta} \mid \{C_R\}_{K_L^\beta} \tag{3}$$

Where α, β are two pre-defined parameters, which ensures that the generated keys are independent,

$$K_i^\alpha = PRF_{K_i}^{(n \to n)}(\alpha),\ i \in \{L, R\}. \tag{4}$$

$$K_i^\beta = PRF_{K_i}^{(n \to n)}(\beta),\ i \in \{L, R\}. \tag{5}$$

Clearly, the rekey messages contain enough information for both children to reconstruct K'. For the left child, it can decrypt C_R from $\{C_R\}_{K_L^\beta}$ using its local key, and then combine it with C_L, which is locally available, to reconstruct K'. Symmetric procedures can be applied for the right child.

Given this key updating algorithm, the following procedure can then be executed to meet the demands for client leaving (Fig. 7):

1) The leaving client or a neighboring node that detects its leaving contacts the server;

2) The server deletes the corresponding leaf node in the key tree, and replaces its parent node by its sibling node. The server also informs the sibling node to remove the key of its original parent from the key set;

3) All the keys of the nodes along the key path for the leaving client need to be updated sequentially. For key K_i, the update is as follows,

$$K_i' = PRF_{C_{LR}}^{(n \to n)}(K_i); \tag{6}$$

4) The server multicasts the rekey message to all the clients, that is,

$$\{PRF_{K_{Li}^\alpha}^{(n \to n_1)}(K_i)\}_{K_{Ri}^\beta} \mid \{PRF_{K_{Ri}^\alpha}^{(n \to n_2)}(K_i)\}_{K_{Li}^\beta}. \tag{7}$$

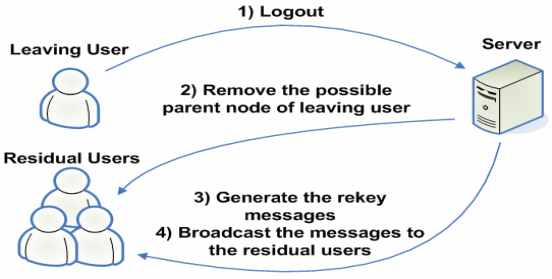


Fig. 7. Procedure for client leave.

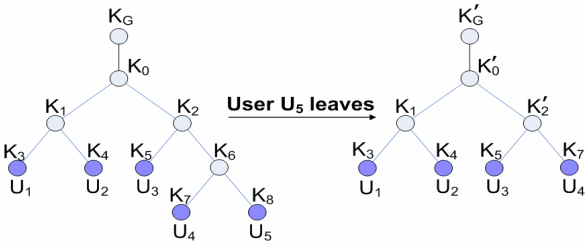


Fig. 8. Illustration of client leaving.

Fig. 8 shows an example for client U_5 to leave the group. It first contacts the server to logout; the server uses its sibling node U_4 to replace their original parent. The key K_6 is then removed from the key set of U_4, and the server computes the new keys along the path from U_4 toward the root. In the last step, rekey messages updating, K_G, K_0, and K_2 are multicast. For illustration, the message for key K_2 is

$$\{PRF_{K_5^\alpha}^{<n \to n_1>}(K_2)\}_{K_7^\beta}, \{PRF_{K_7^\alpha}^{<n \to n_1>}(K_2)\}_{K_5^\beta}. \tag{8}$$

For client U_3, it should first compute $C_{L2} = PRF_{K_5^\alpha}^{<n \to n_1>}(K_2)$, and, upon receiving the right half of this message , that is, $\{C_{R2}\}_{K_5^\beta}$, it decrypts the message to get C_{R_2}. It follows the algorithm in Fig. 6,

$$C_{LR_2} = C_{L_2} \mid C_{R_2} \tag{9}$$

$$K_2' = PRF_{C_{LR2}}^{<n \to n>}(K_2). \tag{10}$$

Considering a balanced binary key tree with N leaves, the number of keys to be updated is bounded by $O(\log N)$, which implies that the leaving algorithm is scalable as well.

When multiply clients leave in a same time slice, the server will firstly gather all of these leaving requests. By the end of this time slice, the server merges the redundant key updating operations, forms an updated key tree and generates rekey messages for all the residual clients to complete their key updating process according to the final key tree. In this case, reduplicate and unnecessary updating of key nodes can be efficiently prevented. Consequently, it can greatly decrease the computational overhead.

V. The Key Embedding Module

The key embedding algorithm of CASM works in the

Discrete Cosine Transform (DCT) domain through a novel Average Luminance Value (ALV) modulation [10]. It partitions the DCT blocks in a frame into mutually-exclusive fields, and alters the Direct Current (DC) component by embedding one bit data in each field. Details of the ALV modulation can be found in [10].

TABLE I
FORMAT OF THE REKEY MESSAGE

20bits	K_i
20bits	K_{Li}
20bits	K_{Ri}
34bits	$\{C_R\}_{K_{Li}^{\beta}}$
34bits	$\{C_R\}_{K_{Ri}^{\beta}}$

In our implementation of CASM, we divide each video frame into 200 fields, and 200 bits data can thus be embedded. The data to be embedded include an 8-bit flag and a 128-bit rekey message, which follows the format presented in Table I. We use the remaining 64 bits for Reed-Solomon (RS) coding, which facilitates client-side error recovery [11]. In addition, the same rekey message is repeated in every subsequent frame till another key is updated, which further enhances reliability of CASM.

VI. THE ENCRYPTION MODULE

We employ a two-level approach for selective encryption, which is the core operation in the secure transmission module. Recently, the H.26x and MPEG1/2/4 video coding standards have provided the enabling technologies for a wide variety of applications. These standards achieve high compression performance using a number of methods to exploit the spatial redundancies and temporal redundancies [23]. The MPEG standard uses inter-frame compression to achieve compression ratio of about 100:1 by storing only the differences between successive frames. The detailed information about video coding can be found in [23]. In the two-level approach, the sign bits of the DC components and motion vectors are respectively modified to encrypt the video data. The modification is an XOR operation with a pseudo-random sequence, which is referred to as an Encryption Sequence (ES), can be calculated as

$$ES = \mathrm{PRF}_{K_G}^{(n\to n)}(frame\_number) \qquad (11)$$

Since the ESs vary with different frame numbers, this scheme can resist statistical attacks. A client, however, keeps track of the session key and the frame number, and thus can easily reproduce the corresponding ES and decrypt the video data accordingly. The detailed encryption operations at each level are given below.

A. Level-1: DC Component Encryptions

At level-1, we shuffle the sign bits of the DC components in an I-frame. Assume that $\overline{DC}_i$ is the codeword to be transmitted for DCT block i, and DC_i is the real DC component. It follows that $\overline{DC}_0 = DC_0$ and $\overline{DC}_i = DC_i - DC_{i-1}$, since the DC components are predicatively coded in the MPEG standard. Let E_i be the *i*-th bit of the ES; we encrypt the DC codeword as follows,

$$Enc(\overline{DC_i}) = (2\times(sign(\overline{DC_i}) \oplus Ki) - 1)\cdot\left|\overline{DC_i}\right|, \qquad (12)$$

where $sign(\overline{DC_i}) = \begin{cases} 0, & if\ \overline{DC_i} < 0 \\ 1, & if\ \overline{DC_i} \geq 0 \end{cases}$.

Given the predictive coding, any change on one DC component will result in severe errors cumulated in all the following blocks. This is illustrated in Fig. 9, where an alternation of the sign in the first block leads to confusion on the luminance components in all the blocks.

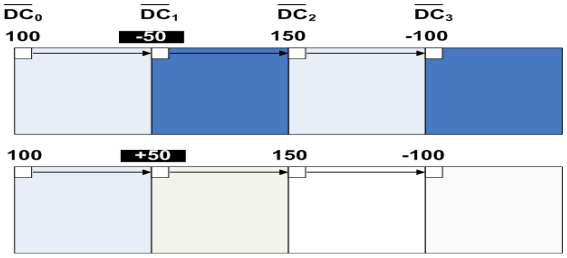


Fig. 9. An example of layer-1 encryption.

B. Level-2: Motion Vector Encryption

It is known that motion vectors contain abundant information about the video. In the scenarios such as video surveillance and conferencing, video objects could be effectively reconstructed by motion vectors only. Hence, it is necessary to hide the motion vectors in a P-frame as well, which is realized in the level-2 encryption.

Assume that MV is the motion vector of the current block, and MV_1, MV_2, MV_3 are motion vectors of its neighboring blocks already being transmitted. Let

$$P_x = Median(MV_{1x}, MV_{2x}, MV_{3x}), \qquad (13)$$

$$P_y = Median(MV_{1y}, MV_{2y}, MV_{3y}), \qquad (14)$$

and the differential codewords for the motion vectors are thus,

$$MVD_x = MV_x - P_x, \qquad (15)$$

$$MVD_y = MV_y - P_y. \qquad (16)$$

Let $E_{(1,t)}$ be the first *t* bits of the ES, where *t* is the length of an MVD. MVDs are encrypted as follows,

$$Enc(MVD_x) = MVD_x \oplus E_{(1,t)}, \qquad (17)$$

$$Enc(MVD_y) = MVD_y \oplus E_{(1,t)}. \qquad (18)$$

Such operations disorder the positions of the reconstructed blocks, leading to unacceptable reconstructed quality for an

entire video frame and, more significantly, all the following predictive frames. An example is shown in Fig. 10. Since the motion vector data are relatively small, the computation overhead of such encryption can be kept at a low level, as shown in the next section.

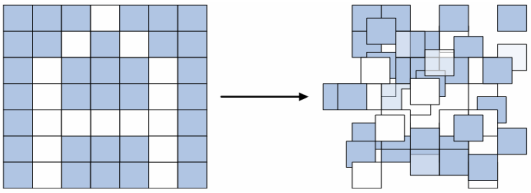


Fig. 10. An example of layer-2 encryption.

VII. Implementation And Experiments

We have implemented a prototype for the proposed CASM system. Its protocol stack is depicted in Fig. 11, and the system configuration is in Fig. 12. Note that we distribute the key management and the key embedding/selective encryption operations into a gateway and a server, respectively. There is also a transcoder that realizes all these functionalities and yet meets the bandwidth constraints from its local clients [2, 6]. Such a configuration validates the flexibility of the modularized design of CASM.

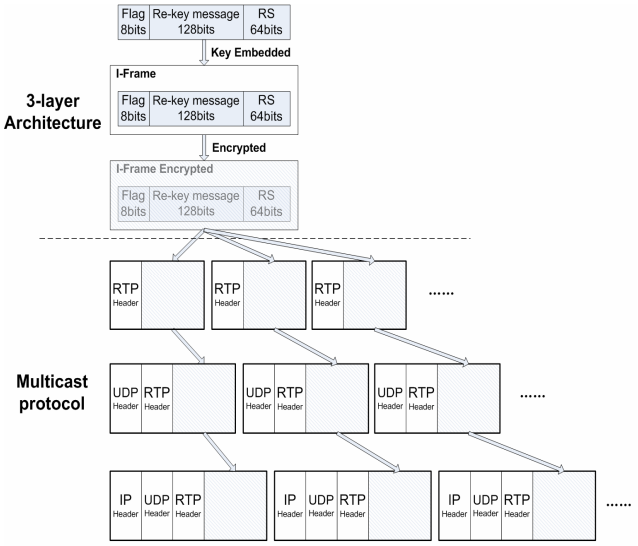


Fig. 11. Protocol stack of the prototype.

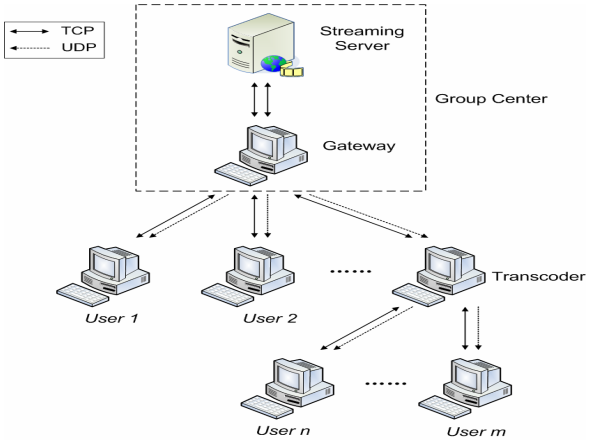


Fig. 12. System configuration of the prototype.

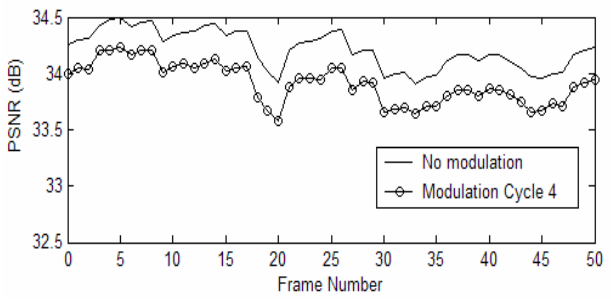
In this prototype, we employ a keyed-hash function, TLS-PRF [24], in the key embedding and selective encryption modules to generate pseudo-random sequences. It splits the input into two halves, transforms them using MD5 and SHA-1, respectively, and then combines them through an XOR operation. Details of TLS-PRF can be found in RFC2246 [24].

A. Reconstruction Quality

In the first set of experiments, we examine the performance of CASM in terms of the quality of reconstructed video. Note that CASM is content-aware, which is reflected in two aspects: first, it embeds the rekey messages into video frames to avoid a separate control channel for key distribution; and second, it encrypts the DC components and motion vectors of a video stream only, which significantly reduces the computation and communication overhead. It is thus necessary to investigate the following two aspects: the encryption effect and the impact on the quality of correctly decrypted content.

We have conducted experiments over a variety of video sequences to investigate the aforementioned two measures. Fig. 13(a) shows a video frame with a 200-bit rekey message embedded (right) and the original frame (left), as well as the corresponding PSNR curves. This sequence containing 500 frames is taken from a famous film “Jurassic Park”. The modulation cycle in the embedding algorithm is 4 (see [10] for details). In this frame, we are unaware of any perceptible quality distortion, which is also true for most of other sequences used in our experiments.

(a)



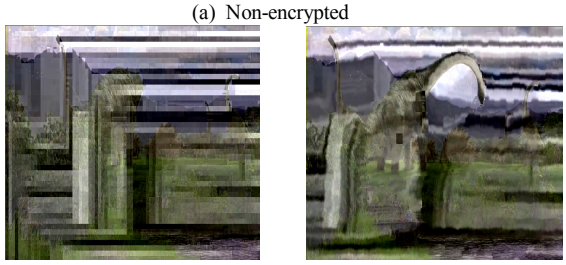
(b)

Fig. 13. (a) Left: un-embedded image; Right: embedded; (b) Corresponding PSNR curves of these two images

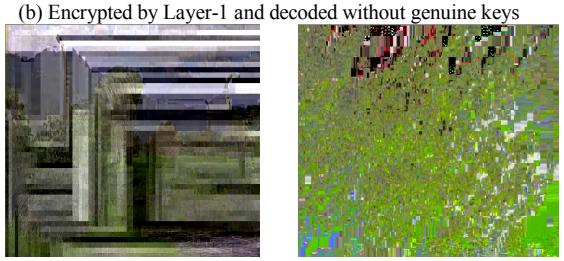
Fig. 14 demonstrates the effectiveness of the two-level encryption algorithm. The original pictures are shown in Fig. 14(a), which also represents the pictures being encrypted and then decoded with a genuine key. In Fig. 14(b), only level-1 encryption is applied and then decoded with an outdated key. We can see there are many stripes, brighter, or darker pixels and blocks in these pictures. While the quality is quite poor, there is still a vague shape of a dinosaur. Upon applying the level-2 encryption for motion vectors, however, it is impossible to recognize the pictures, especially for P-frames (see Fig. 14(c)).

It is worth noting that, for both the key embedding and selective encryption algorithms, there are no extra bits added to the video data, and the altered stream strictly follows the original encoding format. Consequently, no any change is needed for the video decoder on client sides.

(a) Non-encrypted



(b) Encrypted by Layer-1 and decoded without genuine keys



(c)Encrypted by level-1 and -2 and decoded without genuine keys

Fig. 14. Effectiveness of encryption (Left column: I frame; Right column: P-frame).

B. Computation Overhead

The most intensive computation in CASM is required by the selective encryption algorithm, which is to be invoked for each frame. Table II compares the encoding speeds (frame/second) with and without selective encryption for both a high-rate stream (4.5 Mbps) and a mid-rate stream (1.5 Mbps). The results reaffirm that the selective encryption algorithm, applied to a small set of data only, is highly efficient for real-time processing.

TABLE II.
COMPLEXITY OF THE SELECTIVE ENCRYPTION.

Video Sequence	High-Rate	Mid-Rate
Encoding speed without selective encryption (frame/sec)	49.7f/s	55.6f/s
Encoding speed with selective encryption (frame/sec)	48.4f/sec	54.3f/s
Processing time (%)	2.69%	2.39%

C. Scalability

Since the scale of our prototype remains limited, we investigate the scalability of CASM through simulations. Table III shows the client joining and leaving times with different group sizes. The path length is the maximum number of nodes within a root-to-leaf path in the key tree, which is bounded by in CASM. It is easy to show that, when a client joins, only one message should be multicast, and when a client leaves, the number of messages to be multicast ranges from $\lfloor \log_2 N \rfloor + 1$ to $\lceil \log_2 N \rceil + 1$. Considering the multiple concurrent requests, the processing overhead can be greatly decreased by the method of gathering request firstly then processing them later, i.e., a time-driven strategy. Such analysis is consistent with the simulation results presented in Table III, which suggests that CASM can scale to large multicast groups.

TABLE III
JOINING/LEAVING TIME VS GROUP SIZE.

Group size	Path length	Joining time	Leaving time
100	9	0.5s	3.7s
1,000	12	0.5s	5.3s
10,000	16	0.5s	7.4s
100,000	19	0.5s	9.0s

VIII. CONCLUSIONS

This paper has presented a novel content-aware protocol for secure video multicast. The protocol, called CASM, incorporates a light-weight scalable algorithm for group key management, a reliable key embedding algorithm, and a

selective video encryption algorithm. CASM is content-aware, which is reflected in two aspects: first, it embeds the rekey messages into video frames to avoid a separate control channel for key distribution; and second, it encrypts the DC components and motion vectors of a video stream only, which significantly reduces the computation and communication overhead. Moreover, it is fully compatible with existing video coding standards.

We have built a prototype of CASM that demonstrates its robustness and scalability. The design of CASM adopts a time-driven strategy, i.e., dividing time into slices of identical length, which can efficiently accommodate with concurrent multiple requests and can ensure a limited communication overhead. Our experimental results have also validated that CASM enables secure key and video content distribution with minimized bandwidth overheads. Its content-aware key embedding and encryption algorithms are fast enough to support real-time video multicasting.

REFERENCES

[1] J. Liu, B. Li, Y.-Q. Zhang, Adaptive Video Multicast over the Internet, IEEE Multimedia, Vol.10, No.1, January 2003.

[2] P. Yin, M. Wu, and B. Liu, Video Transcoding by Reducing Spatial Resolution, In Proceeding of the IEEE Int'1 Conf. on Image Processing (ICIP), Vancouver, September 2000.

[3] K.-C. Chan, S.-H. G. Chan, Key Management Approaches to Offer Data Confidentiality for Secure Multicast, IEEE Network, pp. 30-39, September 2003.

[4] W. Trappe, J. Song, R. Poovendran, and K. J. Ray Liu, Key Management and Distribution for Secure Multimedia Multicast, IEEE Transaction on Multimedia, Vol. 5, No.4, pp. 544-557, December 2003.

[5] A. M. Alattar, E. T. Lin, M. U. Celik, Digital Watermarking of Low Bit-Rate Advanced Simple Profile MPEG-4 Compressed Video, IEEE Transaction on Circuits and Systems for Video Technology, Vol. 13, No.8, August 2003.

[6] A. Vetro, C. Christopoulos, and H. Sun, Video Transcoding Architectures and Techniques: An Overview," IEEE Signal Processing Magazine, pp.18-29, March 200.

[7] G. Voyatzis and I. Pitas, The Use of Watermarks in the Protection of Digital Multimedia Products, Proceedings of the IEEE, Vol. 87, No. 7, pp.1197-1207, July 1999.

[8] I. J. Cox, J. Kilian, F. T. Leighton and T. Shamoon, Secure Spread Spectrum Watermarking for Multimedia, IEEE Transactions on Image Processing, Vol. 6, No. 12, pp. 1673-1687, December 1997.

[9] S.-J. Lee and S.-H. Jung, A Survey of Watermarking Techniques Applied to Multimedia, In Proceedings of IEEE International Symposium on Industrial Electronics, pp.272-277, June 2001.

[10] H. Yin, X. Chu, C. Lin, F. Qiu, G. Min ,A Novel Key-Embedded Scheme for Secure Video Multicast Systems," Lecture Notes in Computer Science, Springer Vol.3768, pp.246-257,2005.

[11] S. B. Wicker, V. K. Bhargava, Reed-Solomon Codes and Their Applications, Wiley-IEEE, September 1999.

[12] S rafaeli, A Decentralized Architecture for Group Key Management, PhD Thesis, Lancaster University, Lancaster, uk, September 2000.

[13] A. Perrig, D. Song, J. D. Tygar, ELK: A New Protocol for Efficient Large-Group Key Distribution, In Proceedings of the IEEE Symposium on Security and Privacy, 2001.

[14] Y. Li, Z. Chen, S. Tan, and R. Campbell, Security Enhanced Mpeg Player, In Proceedings of IEEE International Workshop on Multimedia Software Development (MMSD'96), Berlin, Germany, March 1996.

[15] T. B. Maples and G. A. Spanos, Performance Study of a Selective Encryption Scheme for the Security of Networked, Real-Time Video, In Proceedings of 4th International Conference on Computer Communication and Network, Las Vegas, NV, September 1995.

[16] J. Meyer and F. Gadegast, Security Mechanisms for Multimedia Data with the Example of MPEG-1 Video, Available at http://www.powerweb.de/phade /phade.html

[17] H. Chu, L. Qiao, and K. Nahrstedt, A Secure Multicast Protocol With Copyright Protection, ACM SIGCOMM Computer Communications Review, Vol. 32, No. 2, April 2002.

[18] L. Tang, Methods for Encrypting and Decrypting Mpeg Video Data Efficiently, In Proceedings of ACM Multimedia'96, Boston, MA, November 1996.

[19] L. Qiao and K. Nahrstedt. Comparison of MPEG Encryption Algorithms, International Journal on Computers and Graphics, Vol. 22, No. 3, January 1998.

[20] L. Qiao and K. Nahrstedt, A New Algorithm for MPEG Video Encryption, In Proceedings of The First International Conference on Imaging Science, Systems and Technology (CISST'97), pp. 21-29, Las Vegas, Nevada, July 1997.

[21] C. Shi and B. Bhargava, A Fast Mpeg Video Encryption Algorithm, In Proceedings of the 6th ACM International Multimedia Conference, Bristol, UK, September 1998.

[22] C. Shi and B. Bhargava, An Efficient Mpeg Video Encryption Algorithm, In Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems, October 1998.

[23] Yao Wang, Jörn Ostermann, and Ya-Qin Zhang, Video Processing and Communications, Prentice Hall, 2002 ISBN 0-13-017547-1

[24] RFC 2246, The TLS Protocol Version 1.0, http://www.faqs.org/rfcs/rfc2246.html

Hao Yin, is an assistant professor of the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He received Ph.D. degree in Information and Communication Engineering from Huazhong University of Science and Technology in 2002. From 2001-2002, he was a CTO with Wuhan Teamswan Digital Technology, Co.,Ltd. His research interests span broad aspects of performance evaluation for Internet and wireless network, image/video coding, multimedia communication over wireless network, and security. He has published more than 50 papers in refereed journal and conference.

Chuang Lin,is a professor and the head of the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He received the Ph.D. degree in Computer Science from Tsinghua University in 1994. His current research interests include computer networks, performance evaluation, logic reasoning, and Petri net theory and its applications. He has co-authored more than 150 papers in research journals and IEEE conference proceedings in these areas and has published three books.

Qiu Feng, received the B.Eng degree (cum laude) from Tsinghua University, Beijing, China, in 2004, and the master candidate, both in computer science. His main research is multimedia security.

Jiangchuan Liu (S'01-M'03) received the B.Eng degree (cum laude) from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from The Hong Kong University of Science and Technology in 2003, both in computer science.
He is currently an assistant professor in the School of Computing Science, Simon Fraser University, BC, Canada, and was an assistant professor at The Chinese University of Hong Kong from 2003 to 2004. His research interests include Internet architecture and protocols, media streaming, wireless ad hoc networks, and service overlay networks. He serves as TPC member for various international conferences, including IEEE INFOCOM and IWQoS. He was Information System Co-Chair for IEEE INFOCOM'04, and a guest-editor for ACM/Kluwer Journal of Mobile Networks and Applications (MONET), Special Issue on Energy Constraints and Lifetime Performance in Wireless Sensor Networks. He was Program Co-Chair of The First IEEE International Workshop on Multimedia Systems and Networking (WMSN'05). He is a member of IEEE and ACM, and an elected member of Sigma Xi.

Geyong Min received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University

of Science and Technology, China, in 1995. He is currently a Senior Lecturer in the Department of Computing at the University of Bradford, United Kingdom. His research interests include Performance Modeling/Evaluation, Computer Networks, Mobile Computing and Wireless Networks, Multimedia Systems, Information Security.

Bo Li, (S'89-M'92-SM'99)received his B. Eng. (summa cum laude) and M. Eng. degrees in the Computer Science from Tsinghua University, Beijing in 1987 and 1989, respectively, and the Ph.D. degree in the Electrical and Computer Engineering from University of Massachusetts at Amherst in 1993. Between 1993 and 1996, he worked on high performance routers and ATM switches in IBM Networking System Division, Research Triangle Park, North Carolina. Since 1996, he has been with the Department of Computer Science, Hong Kong University of Science and Technology. He also holds an adjunct researcher position at the Microsoft Research Asia (MSRA), Beijing, China. His research interests are on adaptive video multicast, packet scheduling and dynamic routing in optical networks, resource management in mobile wireless systems, scheduling and energy efficient routing in ad hoc networks, across layer design for sensor networks, and content distribution and replication. He has published 80 journal papers and held several patents in above areas. He received the Outstanding Oversea Young Scientist Award from Natural Science Foundation of China in 2004. He has been on editorial board for 16 journals and involved in organizing over 40 conferences, esp. IEEE Infocom since 1996. He was the Co-TPC Chair for IEEE Infocom 2004.