Securing Media for Adaptive Streaming

Chitra Venkatramani Peter Westerink Olivier Verscheure
IBM T.J. Watson Research Center
P.O.Box 704
Yorktown Heights, NY 10598
{chitrav,peterw,ov1}@us.ibm.com

Pascal Frossard
Signal Processing Institute
EPFL
Lausanne 1015,Switzerland
pascal.frossard@epfl.ch

ABSTRACT

This paper describes the ARMS system which enables secure and adaptive rich media streaming to a large-scale, heterogeneous client population. The secure streaming algorithms ensure end-to-end security while the content is adapted and streamed via intermediate, potentially untrusted servers. ARMS streaming is completely standards compliant and to our knowledge is the first such end-to-end MPEG-4-based system.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed Systems*

General Terms

Design, Security, Standardization

Keywords

Adaptive, Encrypted, MPEG-4, video server,streaming, Scalability

1. INTRODUCTION

Many enterprises use streaming video to convey news clips or corporate communications to their employees or clients. However, since the networks are based on packet-switching technology which is designed for data communication, achieving efficient distribution of streaming video and multimedia to a wide heterogeneous user population poses many technical challenges. Besides the standard video-over-IP issues, enterprises have additional requirements due to the need to control a shared infrastructure where business media comes first. In addition to challenges in terms of video coding and networking, one of the key requirements for enterprise streaming is clearly posed in terms of security. The video distribution has to be efficient and to adapt to the clients requirements, while at the same time offering a high degree of security through proper authentication, authorization and encryption techniques.

In this paper we describe the ARMS (Adaptive Rich Media Secure) streaming architecture, with a focus on original extensions to the secure-streaming standards. The proposed extensions allow various adaptive multimedia streaming techniques over lossy IP networks, to work in the secure domain. To our knowledge, this is the first such end-to-end MPEG-4-based system, which is also completely standards compliant. Specifically, ARMS addresses the following requirements of an end-to-end enterprise media distribution solution – (i) adaptive, secure and standards-based streaming, (ii) scalable broadcast using application-level or IP multicast, (iii) archive and VoD streaming. The ARMS content-creation, encryption and streaming solutions are being implemented in IBM's Content Manager VideoCharger [2] streaming server product.

The rest of the paper is organized as follows. Section 2 describes the related work. Secure streaming algorithms along with adaptivity are described in Section 3. The architecture of the ARMS system where the algorithms are implemented is described in Section 4. Finally our plans for the future of this project are discussed in Section 5.

2. RELATED WORK

Various coding strategies address the problem of serving heterogeneous clients with adaptive video quality. In *Stream Replication* [4], the source signal is encoded into multiple independent streams, each suited to a set of client characteristics such as available rate, packet-loss characteristics, etc. In *Layered Coding* [6], the video is encoded into multiple layers consisting of one base layer and multiple enhancement layers. Each enhancement layer provides progressive refinement of the signal. MPEG-2 and MPEG-4 define spatial, temporal and SNR scalability modes. In *Multiple Description Coding* [9], the video is encoded in two or more independently decodable layers. The decoded signal quality is proportional to the number of layers decoded. Although MDC has been shown to be a very promising choice for adaptive streaming in lossy environments, the MPEG-4 standard does not support it. Among these schemes, stream replication and layered coding are standards compliant, although layered-coding is not in widespread use. One of the reasons is that for a few targeted bit-rates (which is commonly the case in enterprises), coding individual streams yields better quality than multiple layers [5].

Several adaptive streaming techniques are also in use. The *Stream Switching* mechanism works with stream replication and multiple description coding where the encodings are independent. Depending on the client feedback about the observed packet losses or bandwidth, the server streams the most suitable encoding to the client. In order for the server to do this with minimal distortion due to switching, various strategies have been proposed – SP-frames in H.26L, insertion of periodic I-frames etc. Some solutions, based on layered coding schemes, choose to *add or drop layers* [7, 12] from the stream. When either the client or the server realizes that

MM'03, November 2–8, 2003, Berkeley, California, USA.
Copyright 2003 ACM 1-58113-722-2/03/0011 ...$5.00.

the connection speed has changed, additional video layers are either dropped or added to improve the reception quality. Another alternative is to *add/drop whole streams* such as video in favor of audio, at the server, based on measurements or policy. *On-the-fly Transcoding* [8, 14] has also been suggested where the server transcodes the stream to address the changing network conditions. This method is computationally expensive and may not be very scalable.

Various secure distribution schemes are also in use. Most use the download model, where the content is encrypted at the source, downloaded and decrypted by the client. This method prevents media from being streamed and adapted since the server cannot access the encrypted media. Papers by Wee [11] describe a secure encoding system that is based on scalable encoders which allows for adaptation while still in the secure domain. The technique however, does not apply to non-scalable coding techniques and the encryption and packetization techniques are non-standard.

ARMS uses the stream-replication technique and adopts novel enhancements such that only portions of the stream are multiply encoded. This improves the achievable rate-distortion at the client significantly, while still being network-efficient. The encryption and adaptation techniques described in this paper work regardless of whether the entire stream or only portions are multiply encoded. Data is encrypted and packaged at the source such that the intermediate server can implement a simple and efficient stream-switching algorithm, based on the measured bandwidth to the client. Since the encryption uses a counter-based cipher (See Section 3), it is loss-tolerant and is applicable to both scalable and non-scalabe encoding profiles of MPEG-4, such as layered encoding and MPEG-4 Fine Grained Scalabe (FGS) coding. This scheme may also be applicable to proprietary encoding methods as used by Microsoft or RealNetworks.

3. SECURE STREAMING

This section gives a brief overview of the ISMA security standard [3] and details the enhancements made by us to enable adaptive streaming of secure content. This enables trustworthy distribution of content over potentially untrusted servers. ISMA stands for Internet Streaming Media Alliance and is an industrial consortium, of which IBM is one of the founding members.

3.1 ISMA Security Standard

Streaming media can be delivered as a file download, lossless streaming over TCP, or potentially lossy streaming over UDP. Additionally, the delivery of the protected content need not necessarily be done via trusted intermediaries. Therefore, standardization bodies like the Internet Streaming Media Alliance (ISMA) have opted for an end-to-end security model. As shown in Figure 4, only the content creator, the trusted Key Management System, and the client are in the trusted domain. For untrusted intermediaries to be able to distribute data, the formats standardized are such that sufficient meta data is available at all stages so that delivery can be done without needing access to the actual multimedia content. That is, the encryption is done at the *content* rather than at the transport level, making the protection transport-independent and therefore end-to-end secure.

The ISMA Standard specifies streaming of multimedia over IP networks using RTP over UDP. In such a system, there can be data loss and the amount of data lost is typically unknown. To enable resynchronization of the decryptor with the data under these conditions, we use a stream cipher that allows random access. A stream cipher has an index associated with each byte (or bit) of data. This index is typically the position in the stream of bytes and can be used to start decrypting a stream from that position. Examples of such stream ciphers are SEAL [10] and a block cipher in Counter Mode. Examples of a block cipher are DES [13] and its successor Advanced Encryption Standard (AES) [1]. For an overview on cryptography, see [13].

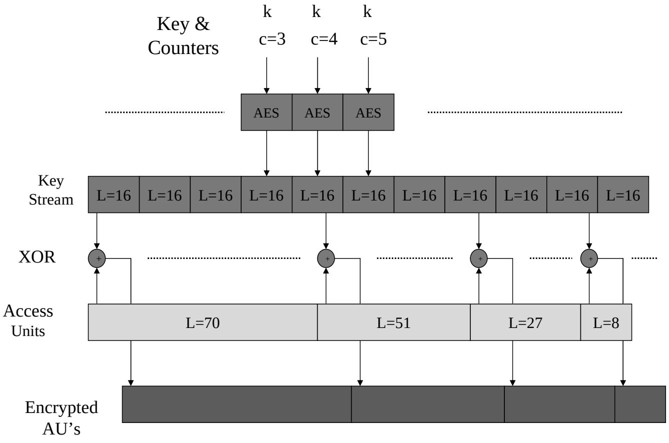

Figure 1: Encryption of a stream of data consisting of 4 Access Units.

For encryption and decryption of streams, we use the AES in Counter Mode, where the counter value is 16 bytes (128 bits). Given a 16, 24, or 32-byte encryption key, a counter value is input into the AES cipher, which produces a 16-byte block. In Counter Mode the data encryption is then performed by bitwise XOR-ing the enciphered counter value with an equally sized data block. To encrypt another 16-byte block of data, the counter is incremented by 1, and the above steps are repeated. To encrypt a continuous stream of multimedia Access Units (such as video frames), we keep incrementing the counter each time we need more data to apply the XOR to. This is shown in the Figure 1 for a stream of 4 Access Units of sizes 70, 51, 27, and 8. The decryption process is identical to the encryption process. The same enciphered counter values are created, but is now XOR-ed with the cipher text to recreate the data. In order to be able to address each data byte by itself, the counter is extended with the 4-bit offset into the 16-byte block to form the encryption index. Given this index, we can restart the decryption at any given point and can therefore have random access into an encrypted stream, and are able to resynchronize decryption after a loss of (any unknown amount of) data.

To deal with data loss, every RTP packet includes the encryption index with the corresponding encrypted data. This index is in the clear, which does not pose a security risk, and can be used to resynchronize the decryption process in case of packet loss. Because the actual index can be quite large, typically only that part of the index that is expected to change to bridge an expected amount of packet loss is included in the RTP packet. For example, if only the 24 least significant bits of the encryption index are included in each RTP packet, then a loss-burst of 2^{24} bytes can be tolerated. If larger bursts of data loss are anticipated or it must be possible to at any time join a broadcast that may run for example for a few hours, then a longer encryption index should be included in RTP packets. The standard allows for the inclusion of a full index (of 132 bits) in any packet in the stream.

3.2 ARMS Adaptation on Encrypted Stream

In ARMS streaming, the media server receives multiple encoded streams and streams a subset to each client depending on their bandwidth characteristics. Besides, the server may switch among streams dynamically depending on the observed bandwidth. Note that the network conditions need not be the only condition the server reacts to. External agents such as policy engines can also influence this decision.

Putting together stream-switching and end-to-end security imposes some constraints on how the encryption index is chosen. As a cryptographic constraint, the same index must never be used twice for the same encryption key. That constraint can be overcome by encrypting the alternate Access Units (streams) with different keys. However, we must also take into account that we do not want to send the complete index (e.g. the 132 bits) by relying on an index that always increases minimally from packet to packet. Sending a complete index would increase the overhead from a mere 16-24 bits to 132 bits per RTP data packet, which can be deemed unacceptable, especially if packets are small. Hence the RTP payload format for encrypted data carries a 16-24 bit index per packet. The length of this index depends on the expected length of a packet-loss burst.

In the ARMS case, we need to tackle the problem of encrypting a multimedia stream that may have multiple Access Units for each time slot. The problem is to select the encryption index for each Access Unit such that the encryption index always increases, regardless of which Access Unit the server selects for transmission. There are two cases to deal with :

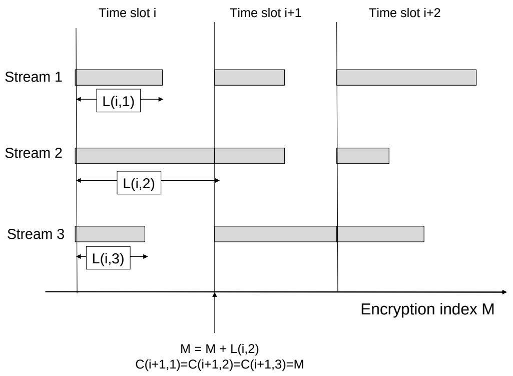

Figure 2: Determination of the encryption index when the key is different in each stream.

- **Each stream has a different key** : In the most common situation a different key is used for each alternate Access Unit for any given time slot. At any given time slot i we shall define that a stream has N(i) Access Units (descriptions), the encryption indexes of those being C(i,1) through C(i,N(i)) and the corresponding sizes being L(i,1) through L(i,N(i)), in bytes. If we represent each Access Unit as a horizontal bar with a size proportional to the length of that Access Unit, the solution can be depicted as shown in Figure 2. Indexes of all the access units at a time slot are equal and the indices at time slot (i+1) is computed as :

$$C(1, i+1) = ... = C(N(i+1), i+1) =$$

$$max(n = 1, , N(i))C(n, i) + L(n, i)$$

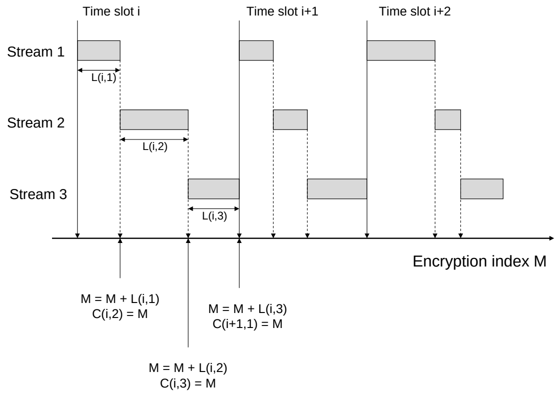

Figure 3: Determination of the encryption index when the key is the same for all the streams.

- **Each stream uses the same key** : In case a single key is used for all streams, then the same index can not be used more than once and we have to increment the index from one Access to the next. This is depicted in Figure 3, where again the sizes of the horizontal bars correspond to the Access Unit lengths. Here the index at time slot (i+1) is computed as below :

$$C(1, i+1) = C(N(i), i) + L(N(i), i)$$

$$C(N(i+1), i+1) = C(N(i+1)-1, i+1) + L(N(i+1)-1, i+1)$$

4. SYSTEM ARCHITECTURE

In this section we describe how the secure streaming technique is used in the end-to-end ARMS system. The main components of the architecture are illustrated in Figure 4. The components consist of the *Broadcaster* which is the source of encrypted content, packaged for adaptation, the *VideoStore* to store the possibly multiply encoded content, the *Streaming Server* which uses a simple and efficient stream-switching technique for adaptation, and finally the playback *Clients*. The figure illustrates a simple configuration with one instance of each of the main components. In large scale deployments, the streaming servers can be networked for distribution and there can be multiple Broadcasters and VideoStores.

Broadcaster : The broadcaster passes raw audio and video input through a bank of MPEG-4 encoders to produce encoded streams in multiple resolutions. These are then passed through an encryption module which keeps all the necessary streaming headers in the clear and only encrypts the content in secure containers. The secured content will then be accessible only by end-clients with a valid key. Finally the data is packetized for transport in the RTP format. The particular payload format used depends on the media type and whether the data is encrypted or not. For non-encrypted data, the standard IETF payload formats are used. When the data is

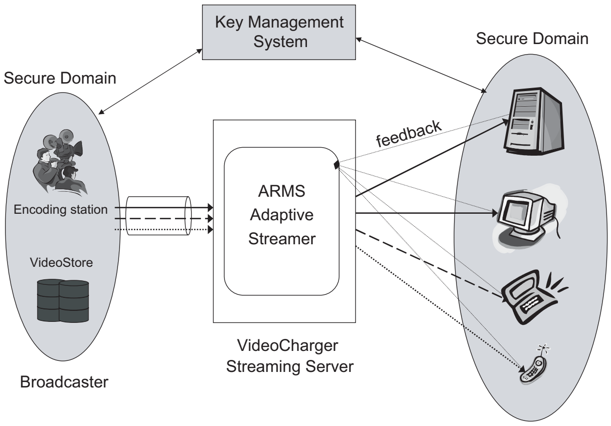

Figure 4: Architecture of a End-to-end Secure Streaming System.

encrypted, the ISMA ”DRM” payload format is used, which, at the time of this writing, is under ISMA-member review. It is expected to be published and presented to the IETF for consideration soon.

VideoStore : This component uses standard ISMA MPEG-4 hinting to store the different stream resolutions under different hint-tracks, each with metadata describing the stream characteristics such as encoding rate, packet-loss sensitivity, etc. The encryption module is then used to encrypt the content into secure containers. Since such content is MPEG-4 compliant, it can be streamed by any compliant streaming server. However, only servers with the ARMS capability will be able to switch among tracks and adapt to changing client conditions.

Streaming Server : The Streaming Server receives data either from the broadcaster or from the videostore. The encryption of the media data is transparent to the server since it only uses information in the hint-tracks or packet headers to stream the data. Data is received from the broadcaster in the form of RTP packets over one of many different transports – (i) multiplexed in one TCP channel in the RTSP-interleave format, (ii) over multiple independent UDP unicast channels or (iii) over multiple independent UDP multicast channels. In each case, a unique channel number identifies packets belonging to a particular encoding. The server measures the available bandwidth to each of its clients and forwards the most suitable channel to the client. For TCP-based client connections, TCP-backpressure is used to estimate the available rate and for RTP/UDP connections, RTCP feedback is used to estimate the TCP-friendly rate. If the streamed bandwidth is well below the rate requested by the client, then the server periodically attempts to increase the bandwidth to the client by probing it with duplicate packets.

Client : The client obtains the SDP describing the media and the encryption keys from a Key Management system. The Videocharger server is capable of streaming MPEG-4 to any standards compliant client such as Apple QuickTime 6, Philips player, Cisco player and the IBM player. Among these, at the time of this writing, only the IBM player implements the ISMA decryption standard.

5. FUTURE WORK

In this paper, we described the ARMS system architecture with a focus on the extensions to the ISMA security standard to enable adaptative streaming of encrypted MPEG-4 content. The system is designed to address the requirements of typical enterprise media streaming systems. Although we have addressed many challenges in building this system, there are many more problems yet to be solved. We are investigating various optimizations in the coding and streaming to improve the bandwidth utilization while minimizing the distortion experienced by the clients in wired and wireless networks.

6. REFERENCES

[1] Advanced Encryption Standard (AES), NIST FIPS 197 http://csrc.nist.gov/encryption/aes/index.html.

[2] IBM Content Manager Videocharger.

[3] Internet Streaming Media Alliance Implementation Specification, Version 1.0. *Internet Streaming Media Alliance*, Aug 2001.

[4] Cheung X.Y., Ammar M.H. and Li X. On the use of destination set grouping to improve fairness in multicast video distribution. *Proceedings of IEEE Infocom*, Mar. 1996.

[5] Kim T. and Ammar M.H. A Comparison of Layering and Stream Replication Video Multicast Schemes. *Proceedings of the NOSSDAV*, June 2001.

[6] V. J. A. McCanne S. and V. M. Receiver-driven layered multicast. In *ACM SIGCOMM*, volume 26,4, pages 117–130, New York, Aug. 1996. ACM Press.

[7] McCanne S., Vetterli M. and Jacobson V. Low-complexity Video Coding for Receiver-driven Layered Multicast. *IEEE Journal on Selected Areas in Communications*, 15(6):983–1001, August 1997.

[8] R. K. Puri R., Lee K.-W. and B. V. An integrated source transcoding and congestion control paradigm for video streaming in the internet. *IEEE Transactions on Multimedia*, 3(1):18–32, 2001.

[9] Reibman A.R., Jafarkhani H., Wang Y., Orchard M.T. and Puri R. Multiple-description video coding using motion-compensated temporal prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(3):193–204, March 2002.

[10] Rogaway P. and Coppersmith D. A Software-Optimized Encryption Algorithm. *Journal of Cryptology*, 11(4):273–287, 1998.

[11] W. S. and A. J. Secure scalable video streaming for wireless networks. *In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2001.

[12] Saparilla D. and Ross K.W. Optimal Streaming of Layered Video. In *Proceedings of the IEEE INFOCOM*, pages 737–746, 2000.

[13] Schneier B. *Applied Cryptography*. John Wiley & Sons, 1996.

[14] Wu D., Hou T., Zhu W., Lee H.-J., Chiang T., Zhang Y.-Q. and Chao H. J. On End-to-End Architecture for Transporting MPEG-4 Video over the Internet. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(6):923–941, 2000.