SECURE SCALABLE STREAMING ENABLING TRANSCODING WITHOUT DECRYPTION

Susie J. Wee and John G. Apostolopoulos

Streaming Media Systems Group
Hewlett-Packard Laboratories, Palo Alto, CA USA

ABSTRACT

We present a method of secure scalable streaming (SSS) that enables low-complexity and high-quality transcoding to be performed at intermediate, possibly untrusted, network nodes without compromising the end-to-end security of the system. SSS encodes video into secure scalable packets using jointly designed scalable coding and progressive encryption techniques. This combination allows downstream transcoders to perform transcoding operations such as bitrate reduction and spatial downsampling by simply truncating or discarding packets, and without decrypting the data. Secure scalable packets have unencrypted headers that can provide hints such as optimal truncation points to downstream transcoders. Using these hints, downstream transcoders can perform RD-optimal transcoding for fine-grain bitrate reduction. The SSS transcoding operation has low complexity and is stateless, so SSS transcoders can support many simultaneous transcoding sessions. SSS works with existing scalable image and video compression standards and systems including Motion JPEG-2000, 3D subband coding, and MPEG-4 FGS.

1. INTRODUCTION

A successful video streaming system must be able to stream video to heterogeneous clients over time-varying communication links in a scalable, efficient, and secure manner. Scalability is needed to enable streaming to a multitude of clients with different device capabilities. Efficiency is needed to maximize the utility of available network and device resources. Security is important to protect content from eavesdroppers. In order to achieve scalability and efficiency in streaming media environments, one must be able to easily adapt or transcode the compressed video stream at intermediate network nodes for particular client capabilities or network conditions. Transcoders perform processing operations on compressed bitstreams; useful network transcoding operations include bitrate reduction, rate shaping, spatial downsampling, frame rate reduction, and changing compression formats.

While network transcoding facilitates scalability and efficiency in video streaming systems, it also presents a number of challenges. First, network transcoding poses a serious threat to the security of the streaming system because transcoding encrypted streams generally requires decrypting the stream, transcoding the decrypted stream, and then re-encrypting the result as shown in Figure 1. Since every transcoder must decrypt the stream, each network transcoding node presents a possible breach to the security of the entire system. Furthermore, while computationally efficient transcoding algorithms have been developed [1, 2, 3, 4], even these improved algorithms are not well-suited for processing multiple streams at intermediate network nodes. In addition to the prohibitive computational requirements needed to hold parallel transcoding sessions, network transcoders would have to maintain state information for each transcoding session.

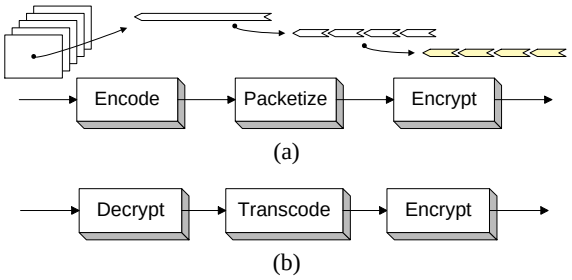

Fig. 1. Conventional (a) secure streaming and (b) transcoding.

In this work, we present a framework for video streaming that simultaneously achieves the three goals of scalability, efficiency, and security despite these challenges. This is accomplished with our proposed method of secure scalable streaming (SSS). SSS encodes video into secure scalable packets that can be streamed across networks to heterogeneous clients. These secure scalable packets can be transcoded to lower bitrates and resolutions at intermediate network nodes with very low-complexity processing and without decryption. Thus, SSS enables low-complexity network transcoding without compromising the end-to-end security of the system.

This paper is outlined as follows. Section 2 describes the fundamental concepts behind SSS and presents a general SSS coder and SSS transcoder. Section 3 discusses properties inherent to SSS transcoding. Section 4 describes how a number of existing scalable video coders and scalable image/video compression standards can be used in an SSS framework.

2. SECURE SCALABLE STREAMING

Our proposed method of Secure Scalable Streaming (SSS) encodes video into secure scalable packets that can be streamed across networks and transcoded at intermediate network nodes with low-complexity processing and without decryption. SSS is based on effectively combining scalable coding and progressive encryption techniques.

2.1. Scalable Coding

Scalable video coding encodes a video sequence into a scalable bitstream with prioritized importance such that the first segment of the scalable bitstream can be used to decode baseline-quality video, and progressively longer segments can be used to decode

progressively improved video. The scalability of the bitstream can have varying degrees of granularity, as coarse as stream-by-stream granularity where layers are placed in different streams, and nearly as fine as bit-by-bit granularity where each additional bit refines the video quality. The video quality can be layered by one or more factors, such as spatial resolution (spatial scalability), bit plane resolution (SNR scalability), or frame rate (temporal scalability). Some image and video compression standards such as JPEG-2000 and MPEG-4 incorporate various forms of scalability and are discussed further in Section 4. Note that when video is coded into a scalable bitstream, lower bitrate or lower resolution bitstreams can be formed by appropriate truncation of the original scalable bitstream.

2.2. Progressive Encryption

If an entire bitstream was encrypted with one long block code, it would not be decryptable unless it was received in its entirety. In order to allow smaller portions of the bitstream to be decryptable, the bitstream could be divided into small blocks which are encrypted independently. While independent block processing is attractive, the use of small block sizes is not very secure. A large degree of security can be added by encrypting small blocks sequentially, and feeding the encrypted data of earlier blocks into the encryption of later blocks as shown in Figure 2. One well-known encryption method that provides this property is called cipher block chaining [5]. In cipher block chaining, a small block of plaintext is first encrypted into ciphertext, this ciphertext is then XORed with the next block of plaintext and the result is encrypted into the next block of ciphertext. This is repeated for the remainder of the plaintext data stream. The resulting decryption operation is also sequential. The first small block of ciphertext is decrypted into plaintext. Then, the second block of ciphertext is decrypted and the result is XORed with the ciphertext from the previous block. This is repeated until the entire message is decrypted. Note that uniform or variable block sizes could be used. For example, larger block sizes could be used at the beginning of the stream and smaller block sizes could be used at the end.

Similar to cipher block chains, stream ciphers [5] also encrypt and decrypt data sequentially. Stream ciphers encrypt plaintext into ciphertext one bit at a time. They use a keystream generator to calculate a stream of bits called a running key. This running key is XORed with the plaintext to produce the ciphertext. Decryption requires calculating the same running key and XORing it with the ciphertext.

We refer to methods that encrypt and decrypt data sequentially as *progressive encryption methods*. Note that when a data stream is encrypted with progressive encryption, earlier portions of the encrypted data can be decrypted even if later portions are not available.

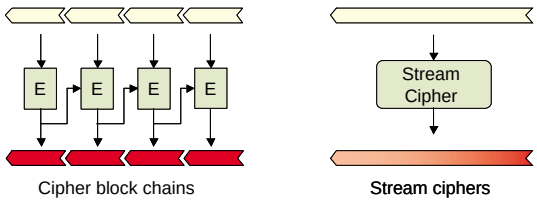

Fig. 2. Progressive encryption methods encrypt and decrypt data sequentially.

2.3. Scalable Coding and Progressive Encryption Enabling Low-Complexity Transcoding

Scalable coding can be used to prioritize or layer data by resolution or SNR. In scalable coding, earlier portions of the scalable stream can be decoded without requiring later portions of the stream. In progressive encryption, earlier segments of ciphertext can be decrypted without requiring later segments of ciphertext. These properties make scalable coding and progressive encryption a nice match for secure scalable streaming. Specifically, by combining scalable coding and progressive encryption as shown in Figure 3a, scalably encoded video may be transmitted across a network in encrypted form; and, downstream transcoders may perform transcoding operations such as bitrate reduction or spatial downsampling by simply truncating the bitstream as shown in Figure 3b. Since the transcoding operation does not require decryption, it does not threaten the end-to-end security of the system.

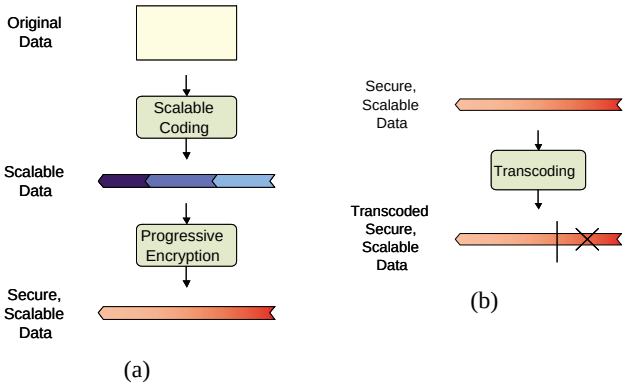

Fig. 3. Appropriate combination of scalable coding and progressive encryption enables transcoding to be performed by simple bitstream truncation.

2.4. SSS Coding and SSS Transcoding

In SSS coding, scalable coding, packetization, and progressive encryption methods are combined to code video into secure scalable packets [6]. It is this combination that allows downstream transcoders to lower the bitrate or spatial resolution of a stream by simply truncating or discarding packets. Secure scalable packets should have a number of characteristics. They must be *scalable* to enable downstream transcoding with packet truncation, *encrypted* to provide end-to-end security, and *independently decodable* to provide resilience to errors such as packet loss.

A basic element of an SSS coder is a scalable coder that encodes video into scalable packets. If the video frame can be encoded into scalable data that fits within a single packet, then many existing scalable image or video compression algorithms can be used directly in an SSS coder. However, if the video frame is too large or has too much activity, then the coded data will not fit into a single packet, so modifications will have to be made.

An SSS coder is shown in Figure 4. First, the input video frame is segmented into regions. Then, each region is coded into scalable video data and header data is formed to convey information to downstream transcoders and decoders. Next, the scalable video data is encrypted with progressive encryption. Finally, se-

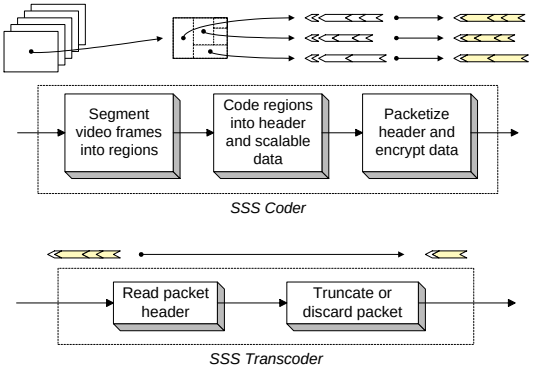

Fig. 4. SSS coders encode video into secure scalable packets. SSS transcoders transcode packets without decryption.

cure scalable packets are created by combining the unencrypted header data with the progressively encrypted scalable video data.

The unencrypted header may describe the region that the packet represents or other information needed for subsequent SSS transcoding and decoding operations. Such information may include a series of recommended truncation points or hints to help downstream transcoders perform transcoding operations such as bitrate or resolution reduction.

An SSS transcoder is shown in Figure 4. SSS transcoders simply read the unencrypted header data at the beginning of each packet, then use that information to discard packets or truncate packets at the appropriate locations. SSS decoders then decrypt and decode the received packets; the resolution and quality of the reconstructed video will depend on the transcoding operation.

An SSS coder may use intraframe or interframe coding techniques. The input of the SSS coder will contain pixel or residual values depending on which is used. If interframe coding is used, a potential issue that may arise is the drift that results if data that was used in the encoder's prediction loop is not available to the decoder because of transcoding. This issue is discussed in [6].

3. SSS TRANSCODING PROPERTIES

3.1. Coarse- and Fine-Grain Transcoding

A straightforward approach to achieving some degree of secure transcoding without decryption involves coding a video sequence into base and enhancement layers, where each layer is coded into separate independently decodable packets which are independently encrypted. The base layer packets can be sent as higher priority packets and the enhancement layer packets can be sent as lower priority packets. An intermediate node may perform secure transcoding by simply discarding the low priority enhancement data packets. This approach provides *secure, coarse-grain transcoding capabilities* since the stream can be transcoded to the two rates that correspond to the low resolution and full resolution bitrates. However, it does not provide finer-grain transcoding capabilities to hit target bitrates that are in between.

SSS allows downstream transcoders to perform bitrate reduction with very fine granularity while maintaining the end-to-end security of the system. This is achieved by appropriately packetizing and encrypting the scalable data. Examples of SSS systems that provide *secure, fine-grain transcoding capabilities* are described in Section 4.

3.2. Secure RD-Optimal Transcoding

A highly desirable feature is the ability to transcode the compressed stream to different rates, each of which is rate-distortion (RD) optimal or near-RD optimal. An overview of RD-optimal coding techniques is given in [7]. Achieving this property is rather straightfoward if an entire frame is coded into a *single embedded bitstream* that is sent within a *single packet*; in this case any truncation point of the single packet (embedded bitstream) will be near-RD optimal by design. However, it is much more difficult when a frame is coded into *multiple packets* and when the packets are encrypted for secure transmission because the transcoder can not see the compressed bitstream.

SSS enables near-optimal transcoding across packets by placing the RD-optimal cutoff points for a number of quality levels in the unencrypted headers of each packet [6]. Then, transcoders can truncate each packet at the appropriate cutoff point for that particular packet; thus, the resulting truncated packets will each contain the optimal number of bits for the new total target bitrate.

3.3. Stateless Transcoding

It should be noted that SSS transcoding is *stateless* and has low complexity. Since transcoding is simply a packet truncation process, downstream transcoders do not have to perform computationally expensive decryption or encryption operations. In addition, they do not have to maintain session state information from packet to packet. For example, a conventional transcoder may need to buffer at least a frame's worth of packets and store session state information from frame to frame. Since SSS transcoding only requires packet truncation, it is stateless; thus, a single SSS transcoder can perform many more simultaneous transcoding sessions. In addition, it allows a video stream's packets to be transcoded at any node and does not require all packets to be processed or routed through a single node. Finally, it should be noted that since the transcoding operation is so simple, packets can be transcoded on very short time scales, so transcoders can quickly react to local network conditions.

4. SSS SYSTEMS

Many types of scalable coders can be used in an SSS system. These coders can use intraframe or interframe coding. They can also use various types of scalability with different levels of granularity. This section describes how a number of existing scalable coders can be used in an SSS framework.

4.1. Motion JPEG-2000 and EBCOT

JPEG-2000 can be used in the proposed SSS framework because it was originally designed with the concepts of tiling (to enable random access) and scalability in mind. JPEG-2000 segments each image into tiles, then codes each tile using SNR or spatially scalable techniques. Secure scalable packets are formed by progressively encrypting the scalable data resulting from each tile and adding the appropriate unencrypted header information.

The JPEG-2000 standard evolved from EBCOT [8] which uses a concept of Post-Compression Rate Distortion (PCRD) optimization to optimally code an image into a bitstream with a desired target bitrate. This is done by gathering RD curve characteristics for

different codeblocks, and coding each codeblock into an embedded bitstream. Specific target bitrates are achieved by extracting the appropriate RD-optimal portions of data from each codeblock and reorganizing these into the final embedded bitstream.

In JPEG-2000 and EBCOT, the codeblock RD information is used to optimally code an image into a desired target bitrate. In SSS, this information is used to calculate truncation points that can be included in unencrypted packet headers to provide hints to downstream transcoders for bitrate reduction. By using this header information, transcoders can perform secure RD-optimal transcoding across packets.

4.2. 3D Subband Coding

3D subband coding can be easily extended into an SSS framework. For example, [9] uses a 3D subband coder to code video into packets such that each packet is independently decodable, of approximately equal importance, and embedded or bitstream scalable. An SSS coder can encrypt the contents of each packet using progressive encryption and thereby enable an intermediate node to perform transcoding by either truncating or discarding packets. Furthermore, recommended trunction points may be placed in the unencrypted header of each packet to enable RD-optimal transcoding across packets.

4.3. MPEG-4 Fine-Grain Scalability (FGS)

MPEG-4 FGS [10, 11] naturally maps into an SSS framework in a number of ways. In FGS, the video is first coded into a base layer with conventional predictive techniques using I, P, and possibly B frames. Note that temporal scalability is inherently provided by using B frames. An enhancement layer is formed by the difference between each coded base-layer frame and the original video frame. This enhancement layer is coded into an embedded FGS bitstream using bitplane coding techniques. The resulting FGS bitstream for each frame can be truncated at any point for bitrate reduction.

An SSS system should encode and encrypt the base and enhancement layers into high priority and low priority packets. The base- and enhancement-layer packets should include unencrypted header information that describes the priority of the packets as hints to downstream transcoders. These downstream transcoders could keep or discard packets based on priority to achieve a coarse level of bitrate scalability.

A note should be made about different methods for packetizing FGS enhancement data. If the FGS enhancement data can be transmitted in a single packet, progressive encryption techniques should be used to encrypt the data so that it can be truncated at any point for fine-grain bitrate reduction by downstream transcoders. If the FGS enhancement data does not fit into a single packet, a number of packetization options are available.

First, the enhancement frame could be coded into one long embedded bitstream, which is then segmented into packets. In this case, packets containing earlier portions of the bitstream are transmitted with higher priority than those containing later portions. Bitrate reduction is then achieved by retaining higher priority packets and discarding lower priority packets to approximately obtain the desired target bitrate. Furthermore, an even finer level of bitrate matching can be achieved by appropriately truncating the packet that straddles the desired target bitrate.

Second, the enhancement data could be partitioned into spatial regions, each of which are scalably coded into separate packets with progressive encryption techniques as shown in Figure 4. In this case, each packet corresponds to a region of the video frame, and can be transcoded by packet truncation. Thus, network-node transcoders can perform fine-grain bitrate adjustment by truncating each secure FGS packet that passes by. The unencrypted header can provide downstream transcoders with near-optimal truncation points for the desired target bitrate.

5. SUMMARY

Secure scalable streaming (SSS) enables downstream transcoding without decryption. SSS uses scalable coding and progressive encryption techniques to encode and encrypt video into secure scalable packets that are streamed across the network. These packets can be transcoded at intermediate, possibly untrusted, network nodes by simply truncating or discarding packets without compromising the end-to-end security of the system. Secure scalable packets have unencrypted headers that can provide hints such as optimal truncation points to downstream transcoders. Using these hints, downstream transcoders can perform RD-optimal transcoding for fine-grain bitrate reduction. Key features of SSS include stateless, low-complexity transcoding; fine-grain bitrate reduction; secure RD-optimal transcoding; and transcoding without decryption. SSS can be used with existing scalable image and video coding standards and systems, including Motion JPEG-2000, 3D subband coding, and MPEG-4 FGS.

6. REFERENCES

[1] H. Sun, W.K. Kwok, and J.W. Zdepski, "Architectures for MPEG compressed bitstream scaling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 2, April 1996.

[2] G. Keesman, R. Hellinghuizen, F. Hoeksema, and G. Heideman, "Transcoding MPEG bitstreams," *Signal Processing: Image Communication*, vol. 8, no. 6, September 1996.

[3] N. Bjork and C. Christopoulos, "Transcoder architectures for video coding," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Seattle, WA, May 1998.

[4] S.J. Wee, J.G. Apostolopoulos, and N. Feamster, "Field-to-frame transcoding with spatial and temporal downsampling," in *IEEE International Conference on Image Processing*, Kobe, Japan, October 1999.

[5] B. Schneier, *Applied Cryptography*, John Wiley & Sons, Inc., 2 edition, 1995.

[6] S.J. Wee and J.G. Apostolopoulos, "Secure scalable video streaming for wireless networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City, UT, May 2001.

[7] A. Ortega and K. Ramchandran, "Rate-distortion techniques in image and video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, November 1998.

[8] D.S. Taubman, "High performance scalable image compression with EBCOT," *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp. 1158–1170, July 2000.

[9] W. Tan and A. Zakhor, "Real-time internet video using error resilient scalable compression and TCP-friendly transport protocol," *IEEE Trans. Multimedia*, pp. 172–186, June 1999.

[10] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301–317, March 2001.

[11] H.M. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 53–68, March 2001.