Secure Scalable Streaming and Secure Transcoding with JPEG-2000

Susie Wee, John Apostolopoulos
Mobile and Media Systems Laboratory
HP Laboratories Palo Alto

HPL-2003-117
June 13th , 2003*

secure scalable streaming, secure delivery, end-to-end encryption, transcoding without decryption, JPEG-2000

Secure scalable streaming (SSS) enables low-complexity, high-quality transcoding at intermediate, possibly untrusted, network nodes without compromising the end-to-end security of the system [1,2]. SSS encodes, encrypts, and packetizes video into secure scalable packets in a manner that 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 provide hints such as optimal truncation points to downstream transcoders. Using these hints, downstream transcoders can perform near-optimal secure transcoding. This paper presents a secure scalable streaming system based on motion JPEG-2000 coding with AES or triple-DES encryption. The operational rate-distortion (R-D) performance for transcoding to various resolutions and quality levels is evaluated, and results indicate that end-to-end security and secure transcoding can be achieved with near R-D optimal performance. The average overhead is 4.5% for triple-DES encryption and 7% for AES, as compared to the original media coding rate, and only 2-2.5% overhead as compared to end-to-end encryption which does not allow secure transcoding.

* Internal Accession Date Only Approved for External Publication
To be published in and presented at IEEE International Conference on Image Processing (ICIP) 14-17 September 2003, Barcelona, Spain

IEEE International Conference on Image Processing, September 2003.

SECURE SCALABLE STREAMING AND SECURE TRANSCODING WITH JPEG-2000

Susie Wee and John Apostolopoulos

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

ABSTRACT

Secure scalable streaming (SSS) enables low-complexity, high-quality transcoding at intermediate, possibly untrusted, network nodes without compromising the end-to-end security of the system [1, 2]. SSS encodes, encrypts, and packetizes video into secure scalable packets in a manner that 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 provide hints such as optimal truncation points to downstream transcoders. Using these hints, downstream transcoders can perform near-optimal secure transcoding.

This paper presents a secure scalable streaming system based on motion JPEG-2000 coding with AES or triple-DES encryption. The operational rate-distortion (R-D) performance for transcoding to various resolutions and quality levels is evaluated, and results indicate that end-to-end security and secure transcoding can be achieved with near R-D optimal performance. The average overhead is 4.5% for triple-DES encryption and 7% for AES, as compared to the original media coding rate, and only 2–2.5% overhead as compared to end-to-end encryption which does not allow secure transcoding.

1. INTRODUCTION

A successful media delivery system should be able to deliver media streams to a multitude of clients with diverse device capabilities connected over heterogeneous networks with possibly time-varying bandwidths. This may require mid-network nodes to perform stream adaptation, or transcoding, to adapt streams for downstream client capabilities and time-varying network conditions [3, 4]. Another important property is security to protect content from eavesdroppers. This makes it necessary to transport streams in encrypted form. In this context, conventional mid-network transcoding poses a serious security threat because it requires decrypting the stream, transcoding the decrypted stream, and 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. Thus, this is not an acceptable solution in situations that require end-to-end security.

Secure scalable streaming is a technology designed to enable the seemingly conflicting goals of end-to-end delivery of encrypted streams and mid-network transcoding without decryption [1, 2]. In other words, it simultaneously enables end-to-end security and secure mid-network transcoding.

In prior JPEG-2000 work, image streaming systems were designed and developed for remote browsing of images [5, 6]. These systems use the scalability and random access features of the standard to provide interactive browsing and adaptability for network conditions and client capabilities. A video streaming system was designed for motion JPEG-2000 [7]. This system uses quality layer scalability to adapt to changing network conditions. All of these systems focus on efficient server-client streaming.

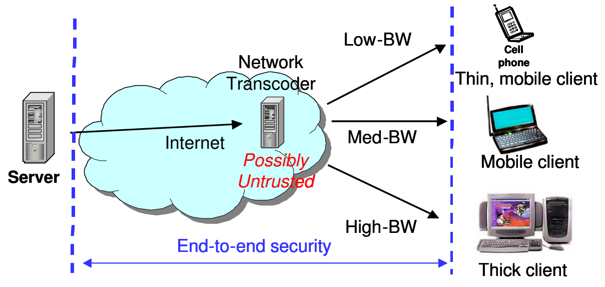


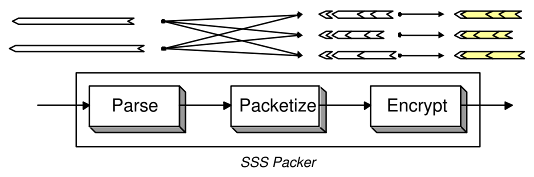
Fig. 1. Challenge of transcoding while preserving end-to-end security: The media should be encrypted at the sender and decrypted only at the receiver – it should remain encrypted at all points in-between. In conventional systems, network transcoding poses a potential security threat because it requires decryption.

In this paper, we present a secure scalable streaming system based on motion JPEG-2000. Similar to the systems presented in [5, 6, 7], our system enables efficient server-client streaming. In addition, it simultaneously provides the seemingly conflicting goals of mid-network transcoding and end-to-end security. This paper begins by providing an overview of secure scalable streaming (SSS). It then presents the details of the SSS system designed for JPEG-2000 and AES and 3DES encryption. It concludes with experimental results that show the rate-distortion performance of the system.

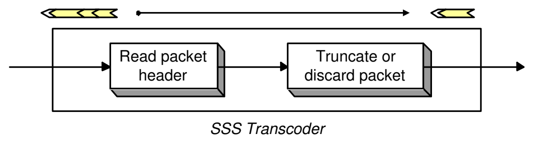
2. SECURE SCALABLE STREAMING (SSS)

SSS is based on an effective combination of scalable coding and progressive encryption. *Scalable coding* methods encode media into scalable data with prioritized importance in a manner that allows the quality of the decoded media to depend on the amount of decoded data. We define the term *progressive encryption* to represent encryption methods that encrypt plaintext into ciphertext in a sequential or beginning-to-end manner [1, 2]; this results in progressive decryption of the resulting ciphertext. By combining scalable coding and progressive encryption, SSS creates secure scalable data that can be transcoded with a truncation operation. This has the benefit of enabling transcoding without decryption. The resulting transcoded secure scalable data can be decrypted and decoded by an SSS receiver with a reconstructed quality that corresponds to the amount of received data. Thus, SSS allows the two seemingly conflicting properties of end-to-end encryption and mid-network transcoding to coexist.

SSS can be used for streaming media over packet networks; however, special consideration must be given to the packetization operation. When used in the context of packet networks, SSS senders must create secure scalable packets that are sent across the network. These packets should be created in a way that allows mid-network SSS transcoders to transcode secure scalable packets with a packet



(a)



(b)

Fig. 2. (a) SSS packers create secure scalable packets from scalably coded bitstreams. (b) SSS transcoders perform secure transcoding, i.e., transcoding without decryption.

truncation or discard operation, without decrypting the data. These resulting truncated packets can then be sent to SSS receivers that can decrypt and decode the data with a quality that corresponds to the amount of received data. The issue that remains is how to create secure scalable packets.

Secure scalable packets are created from scalably compressed bitstreams as shown in Figure 2a. SSS packers parse the scalable bitstream, place the data into packets, and encrypt the result. The scalable data is deliberately placed into packets in a prioritized manner so that transcoding can be performed with packet truncation. This is achieved by placing higher priority data in earlier portions of packets and lower priority data in later portions. Furthermore, SSS packers add unencrypted headers that contain recommended packet truncation points. These unencrypted headers are used by downstream SSS transcoders to guide the transcoding operation.

Mid-network SSS transcoders can adapt secure scalable packets for downstream client capabilities and network conditions. SSS transcoders simply read the unencrypted header data at the beginning of each packet and then discard or truncate packets at the appropriate locations as shown in Figure 2b. SSS transcoders achieve secure transcoding since they perform the transcoding operation without decryption. Furthermore, the SSS transcoder can perform near rate-distortion optimal transcoding *across multiple packets* by adapting the truncation of each packet based on the recommended truncation points contained in the unencrypted header of each packet [1, 2]. Note that the transcoder does not require knowledge of the specific compression algorithm, media type, or encryption algorithm.

Figure 3 shows a more detailed view of the SSS parsing, packetization, and encryption process in the context of the SSS packer. First, original scalable bitstreams are parsed and scalable data segments are extracted for placement in scalable packets. The extracted segments are concatenated to form scalable packet data. A random initialization vector (IV) is used to encrypt each packet to prevent known-plaintext attacks, and padding is appended to the end of the packet to make the packet length amenable to the encryption method. This concatenated data is then progressively encrypted to produce secure scalable data. Associated header data is created to describe features of the secure scalable data, such as recommended packet truncation points or prioritization information that indicates the relative importance of the packet. This header data is left unencrypted so that it can be used by downstream SSS transcoders. Finally, this unencrypted header is then concatenated with the secure scalable data to form a secure scalable packet.

SSS allows streaming media systems to simultaneously achieve two seemingly conflicting properties: network-node transcoding and

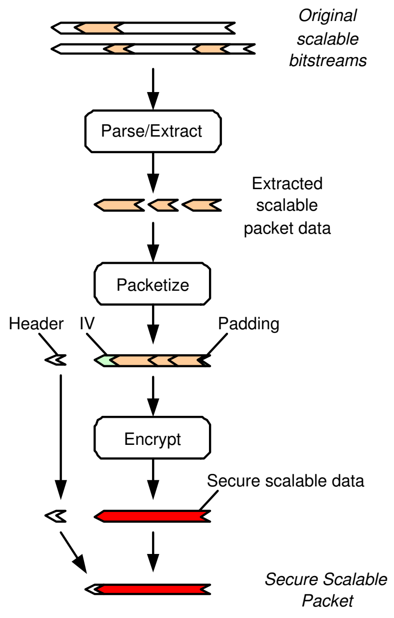


Fig. 3. Formation of a Secure Scalable Packet: A detailed view of the combined packetization and encryption operations.

end-to-end security. Key features of SSS include stateless, low-complexity transcoding; fine-grain bitrate reduction; secure near-RD-optimal transcoding; and transcoding without decryption [1, 2]. 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; and with existing encryption methods, including Data Encryption Standard (DES), Triple-DES (3DES) and the new Advanced Encryption Standard (AES).

3. SSS SYSTEM DESIGN WITH JPEG-2000

We designed and implemented an SSS system using motion JPEG-2000 coding with AES encryption and 3DES encryption. Each frame of a video sequence was coded with a JPEG-2000 coder into a JPEG-2000 scalable bitstream. Each bitstream was parsed, packetized, and encrypted into a number of secure scalable packets using the SSS packer shown in Figure 2a. The secure scalable packets were streamed to an SSS transcoder, which received the secure scalable packets, transcoded them with a packet truncation operation, and sent the truncated secure scalable packets to an SSS receiver. By varying the degree of transcoding, the decrypted and decoded packets produced high-resolution, medium-resolution, or low-resolution video frames with varying degrees of quality. Figure 4 shows low-, medium-, and high-resolution video frames that were decrypted and decoded with this system, which is described in detail next.

The JPEG-2000 coding standard allows many modes of scalability. While a number of JPEG-2000 coding settings can be used, in this particular experiment the scalable ordering was set to PCRL (precinct-component-resolution-layer). Three resolution levels were used. The precinct sizes were set to 128x128, 64x64, and 32x32, for the three resolutions resulting in the same number of precincts for each resolution. Experiments were performed using three and nine quality layers. Start-of-packet (SOP) headers were used in the JPEG-2000 coder to mark the beginnings of JPEG-2000 data packets in the coded bitstream (terminology note: JPEG-2000 data packets are different from secure scalable packets). This resulted in bitstreams of the form shown in Figure 5. The resulting bitstreams were fed into

(a)

(b)

(c)

Fig. 4. Received SSS images with (a) no transcoding, (b) transcoding by truncating three segments per packet, (c) transcoding by truncating six segments per packet. Each packet contained nine scalable segments with three resolution levels and 3 quality layers.

an SSS packer with the architecture shown in Figure 2a.

The SSS packer produced secure scalable packets from the JPEG-2000 bitstreams. Specifically, the SSS packer first parsed the JPEG-2000 header to extract the coding parameters used in the bitstream. The extracted parameters included image size, data ordering (e.g., PCRL), number of resolutions, precinct sizes, and number of quality levels. It then placed the data packets (marked by the SOP headers) into secure scalable packets. Since PCRL ordering was used, the data packets in the JPEG-2000 bitstream were ordered such that consecutive data packets corresponded to increasing quality layers of a particular precinct, color component, and resolution; and consecutive groups of data packets corresponded to increasing resolution information of a particular precinct and color component.

In coding setting 3R×3L, the JPEG-2000 coder used three resolution levels and three quality layers per resolution. The SSS packer placed nine consecutive data packets into each secure scalable packet as shown in Figure 5. Each resulting packet contained all the resolution levels and embedded quality layers for a particular precinct and color component. This data was then padded, encrypted, and packetized to form a valid secure scalable packet that could be transcoded by a packet truncation operation as shown in Figure 2b.

In the second coding setting, the JPEG-2000 coder used three resolution levels and nine quality layers. The SSS packer placed 27 data packets into each secure scalable packet. These data packets contained all the resolution levels and quality layers of a particular precinct and color component. In this experiment, the data was ordered so that all the resolution components of a particular quality layer were grouped and placed adjacent to one another, and these groups were ordered into secure scalable packets by increasing quality layer. However, only nine truncation points were specified in the header to mark the beginning of each quality layer. This second coding setting is referred to as 1R×9L. This data was padded, encrypted, and packetized to form secure scalable packets.

We can achieve progressive encryption using a number of approaches [1, 2]. In these tests we use a block cipher and apply it

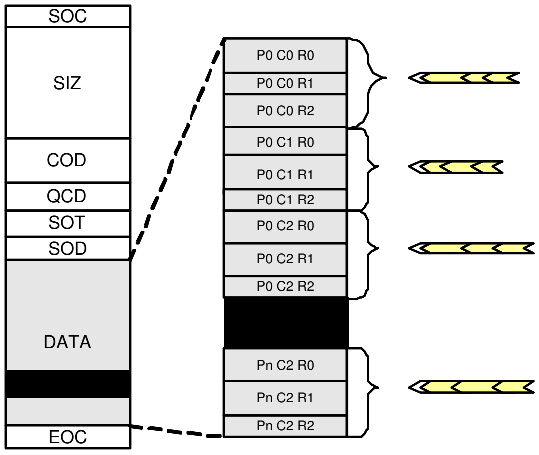


Fig. 5. A JPEG-2000 bitstream and a sample packetization. Each precinct-component-resolution can contain a number of embedded quality layers.

in Cipher Block Chaining (CBC) mode, where previously encrypted data are fedback and used to encrypt the current data. Important examples of block ciphers that provide progressive encryption are DES, 3DES, and the new AES. These are symmetric (secret key) block ciphers that operate on 8-, 8-, and 16-byte blocks using cipher keys of lengths 56, 168, and (typically) 128 bits, respectively. For each of the coding settings discussed above we examined two forms of progressive encryption, based on AES and 3DES. In each case we used an IV of one blocksize, and each packet was padded so that its length was an integer multiple of the blocksize.

4. EXPERIMENTAL RESULTS

The secure scalable packets were sent to a transcoder that performed near R-D optimal transcoding across multiple packets by adapting the truncation of each packet based on the recommended truncation points contained in the unencrypted header of each packet. Experimental results are plotted for the different coding and encryption settings used in our SSS system. Figure 6 shows the performance when the JPEG-2000 coder coded the video frames into three resolutions and three quality layers per resolution, and the SSS packer created packets with nine segments (3R×3L) per packet. Figure 7 shows the performance when coding the video frames into three resolutions and nine quality layers, however the SSS packer was set to create packets with the full resolution and nine quality layers (1R×9L). In both figures, plots (a) and (b) show the results with AES encryption and plots (c) and (d) show the results with 3DES encryption.

Two plots are shown for each coding/encryption setting: an operational R-D plot on the left and an overhead plot on the right. The operational R-D plots show the distortion in mean-squared error (MSE) as a function of the received bit rate in bits per pixel (bpp). Since the coding produces nine resolution/layer segments per packet, each R-D curve has nine points which represent the rate/distortion pairs that result from various degrees of transcoding. Three R-D curves are shown per plot, the first is the R-D curve for the original scalable coded bitstream. The second and third R-D curves plot the R-D values when packing the bitstream into scalable packets and secure scalable packets, respectively.

The overhead plots show the extra bit rate needed for scalable packets and for secure scalable packets as a function of the number of retained packet segments. The overhead is plotted as a percentage over the bit rate of the original scalable coded bitstream. For

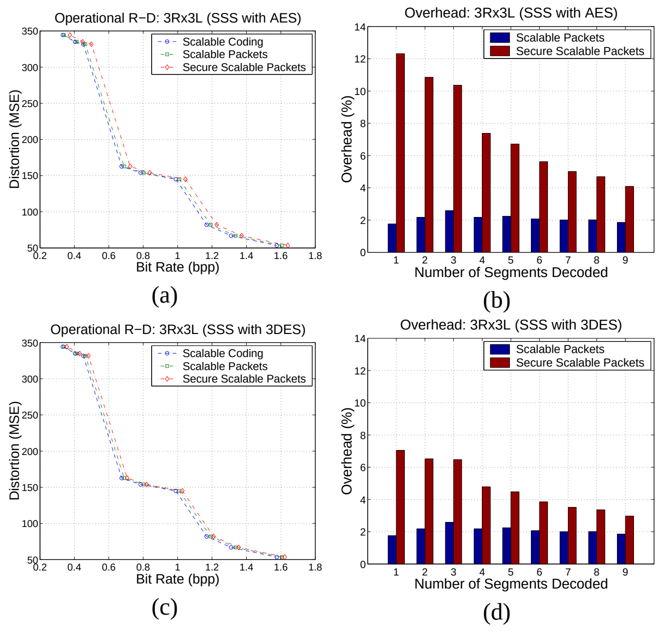


Fig. 6. Experimental results for 3R×3L.

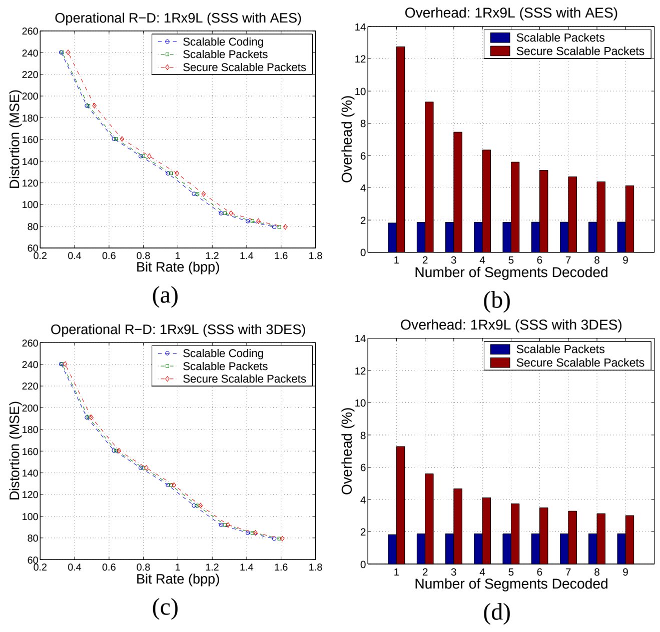


Fig. 7. Experimental results for 1R×9L.

scalable packets, the overhead is due to the extra header information needed to represent the packet truncation points. For secure scalable packets, the overhead is due to truncation points and segment offsets as well as additional overhead needed for the encryption process, specifically, for the initialization vector and padding.

We now comment on the results. First, the R-D curves of the 3R×3L and 1R×9L R-D plots are different in nature. The 3R×3L curves are monotonically decreasing, but have a stairstep type pattern with a step for each resolution level. The 1R×9L curves have a much smoother monotonically decreasing pattern. Also, the 3R×3L curves have a higher maximum distortion and lower minimum distortion over the same range of bitrates. These behaviors are purely a function of the JPEG-2000 coder. Note that the encryption and packetization operations are largely agnostic to the details of the scalable coder, but simply rely on the segment sizes or truncation points of the packetized data.

Second, the distortion levels in the R-D plots are purely a function of the scalable coder, while the rate needed to achieve each distortion level depends on the overhead for each method. Therefore, the three R-D curves in each R-D plot are shifted versions of one another, with shifts that correspond to the extra bit rate needed for the formation of scalable packets and secure scalable packets.

Finally, the overhead for AES is higher than that for 3DES, due to the larger block size used in AES. Therefore it is desirable to use a cipher with a relatively small block size, and of course a large key for security. Also, note that the percentage overhead for scalable packets remains relatively (but not strictly) constant, while the overhead for secure scalable packets decreases with more segments. This is because the overhead for scalable packets is due to the information needed to represent the truncation points of each segment, and thus the extra needed bit rate is proportional to the number of segments. For secure scalable packets, the overhead is due to both the bit rate needed to represent truncation points, and also the extra bits needed for the IV, offsets for each segment, and final padding. The IV and padding are independent of the number of segments per packet, and the offsets which are dependent require very few bits, therefore the additional overhead for security is largely independent of the number of segments. For this reason, the percentage overhead decreases for longer packets (with more segments) since the nearly constant overhead is amortized over the length of the packet. An important note is that a large fraction of the overhead in secure scalable packets is for the IV and padding required for encryption, and that SSS requires only 2–2.5% overhead as compared to conventional end-to-end encrypted delivery which does not allow secure mid-network transcoding.

5. SUMMARY

SSS allows streaming media systems to simultaneously achieve two seemingly conflicting properties: network-node transcoding and end-to-end security. This paper presented an SSS system based on motion JPEG-2000 coding, and AES and 3DES progressive encryption. Performance results show that secure transcoding can be achieved with an average overhead of 4.5% for 3DES and 7% for AES encryption, as compared to the original media coding rate, and only 2–2.5% overhead as compared to end-to-end encrypted delivery which does not allow secure mid-network transcoding.

6. REFERENCES

[1] S.J. Wee and J.G. Apostolopoulos, "Secure scalable video streaming for wireless networks," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2001.

[2] S.J. Wee and J.G. Apostolopoulos, "Secure scalable streaming enabling transcoding without decryption," *IEEE International Conference on Image Processing*, October 2001.

[3] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: An overview," *IEEE Signal Processing Magazine*, March 2003.

[4] S.J. Wee, B. Shen, and J.G. Apostolopoulos, "Compressed-domain video processing," *HP Labs Technical Report 2002-282*, Oct 2002.

[5] D. Taubman, "Remote browsing of JPEG-2000 images," in *IEEE International Conference on Image Processing*, Rochester, NY, September 2002.

[6] S. Deshpande and W. Zeng, "Scalable streaming of JPEG-2000 images using hypertext transfer protocol," in *ACM Multimedia*, October 2001.

[7] R. Qiu and W. Yu, "An efficient quality scalable motion-JPEG2000 transmission scheme," Tech. Rep. WUCS-01-37, Department of Computer Science, Washington University in St. Louis, November 2001.