SECURE MEDIA STREAMING & SECURE ADAPTATION FOR NON-SCALABLE VIDEO

John G. Apostolopoulos

Streaming Media Systems Group
Hewlett-Packard Labs, Palo Alto, CA
japos@hpl.hp.com

ABSTRACT

Two important capabilities in media streaming are (1) adapting the media for the time-varying available network bandwidth and diverse client capabilities, and (2) protecting the security of the media. Providing both end-to-end security and adapting at a (potentially untrusted) sender or mid-network node or proxy can be solved via a framework called Secure Scalable Streaming (SSS) which provides the ability to transcode the content without requiring decryption. In addition, this enables secure transcoding to be performed in a R-D optimized manner. The original SSS work was performed for scalably coded media. This paper examines its potential application to non-scalable media. Specifically, we examine the problems of how to scale non-scalable H.264/MPEG-4 AVC video and how to do it securely. We first show, perhaps surprisingly, that the importance of different P-frames in a sequence can vary by two orders of magnitude. Then we propose two approaches for securely streaming and adapting encrypted H.264 video streams in an R-D optimized manner using (1) Secure-Media R-D Hint Tracks, and (2) Secure Scalable Packets. While we can not scale the bit rate of encrypted non-scalable H.264 to the same extent possible for scalably coded media, our method does provide some scaling capability and more importantly provides 4-8 dB gain compared to conventional approaches.

1. INTRODUCTION

Two important desired capabilities for media streaming are media transcoding or adaptation and end-to-end security, and an important challenge lies in simultaneously enabling both capabilities. For example, it is beneficial to be able to efficiently stream and adapt encrypted media content at potentially untrusted nodes without breaking the end-to-end security. This may be desirable at a potentially untrusted (or vulnerable) streaming server, as shown in Figure 2, or at a potentially untrusted mid-network node or proxy which may need to adapt the incoming media stream to match the down-stream network and client capabilities, as shown in Figure 3. To maximize the security the media should be encrypted by the content creator and decrypted by the content consumer, and everywhere in-between the media should remain in encrypted form, referred to as end-to-end security. The conventional approach for transcoding in the network poses a security threat because transcoding encrypted streams would require giving the node the key, decrypting the stream, transcoding the decrypted stream, and re-encrypting the result – an unacceptable solution as it breaks the end-to-end security.

The author would like to thank Susie Wee, Wai-tian Tan, and Mitchell Trott of the HP Labs Stream Team for their helpful and fun discussions.

The above problem is solved via a framework referred to as *Secure Scalable Streaming* (SSS) [1, 2], which supports end-to-end delivery of encrypted media content while enabling adaptive streaming and transcoding to be performed at intermediate, possibly untrusted, nodes without requiring decryption and therefore without compromising the end-to-end security of the system. We refer to this capability as *Secure Transcoding* to stress that the transcoding is performed without requiring decryption and therefore preserving the end-to-end security. SSS encodes media into secure scalable packets using jointly designed scalable coding, progressive encryption, and packetization techniques. This combination allows potentially untrusted nodes 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 R-D hints such as optimal truncation points which down-stream transcoders can use to perform R-D-optimized transcoding. The SSS framework can in principle be used with any scalable media coder, e.g. speech, audio, image, video. For example, SSS has been examine in the context of JPEG-2000 coded images [3], and is being standardized as part of JPEG-2000 Part-8 Security (JPSEC) [4] (JPSEC supports a number of security services, e.g. [5]). SSS builds on well-studied cryptographic primitives, such as Advanced Encryption Standard (AES) for encryption, and the novelty lies in using these cryptographic primitives in a different manner from how they have been used before.

The prior work has largely focused on secure streaming and secure transcoding for scalably coded content. While it is evident that the SSS framework is applicable to any scalable coder, in this paper we show its applicability to non-scalable coders. This perhaps non-intuitive capability essentially results because all (lossy) media coders inherently produce compressed bits where some bits are more important than other bits. This suggests the idea of reducing the bit rate by deliberately dropping the less important bits.

To examine SSS with a non-scalable coder, we consider the newest video compression standard H.264/MPEG-4 Part 10 Advanced Video Coding (AVC). Specifically, we consider when the video is coded with an initial I-frame followed by all P-frames, and no B-frames. Since the coded video consists of all P frames, it does not suggest a natural prioritization of frames (besides for the earlier P-frames being more important than the later ones). Nonetheless, we show that we can prioritize different P-frames in a surprisingly beneficial manner. We examine two specific examples of SSS with non-scalable H.264: (1) the use of a Secure Media R-D Hint Track for secure adaptive streaming (an extension of [6]), and (2) the use of Secure Scalable Packets for secure mid-network adaptation [1, 2]. In each case the R-D information for each frame or packet is derived and left unencrypted to enable the efficient R-

0-7803-8554-3/04/$20.00 ©2004 IEEE.

D optimized streaming and adaptation, while the coded media data is encrypted.

The closest related work that we are aware of is the recent [7] where the streaming server switches between multiple encrypted copies of the same content compressed at different bit rates (multiple file switching) to adapt to the available bandwidth. However, that work is limited to multiple-stream switching and does not consider the case of adapting a single compressed and encrypted stream as discussed in [1, 2, 3], nor does it consider adapting the stream at a mid-network node.

2. SSS AND NON-SCALABLE VIDEO

In SSS, adaptation is performed while preserving the end-to-end security by viewing the adaptation operation as an intelligent (R-D optimized) select/discard/truncate operation. In addition to avoiding decryption, this approach has the additional benefit of requiring low-complexity. The key steps in performing secure streaming and secure transcoding or adaptation of media are:

1. Understand/analyze the coded media
2. Create (unencrypted) R-D information for maximizing the quality of streaming and transcoding the encrypted media
3. Encrypt media to facilitate easy access
4. Organize/packetize to enable easy access

The above approach is equally applicable for encrypted or unencrypted content, where for unencrypted content the third step (encryption) is not performed. For unencrypted content it provides the benefits of low-complexity R-D optimized streaming and adaptation. Furthermore, the fact that this approach is generically applicable to both encrypted and unencrypted content can be quite useful. The R-D information may be produced during encoding, or can also be derived from pre-encoded content.

Different applications require different levels of security, and in some applications it is important to consider the potential leakage of information. With SSS, the leakage is given by the unencrypted R-D information, and of course by the attributes of the encrypted data itself, e.g. encrypted packet sizes. Therefore, when designing the R-D information it may be useful to consider not only performance and size, but also limiting the potential leakage.

The two basic questions for non-scalable video are: (1) How do we scale non-scalable video? (2) How do we do it securely? Scaling in this context corresponds to reducing the bit rate, or reducing the packet rate or frame rate. Clearly, non-scalable media can not be scaled to the same extent as scalable media. However, it is interesting to understand to what extent it can be scaled.

3. BIT RATE SCALING OF NON-SCALABLE H.264 VIDEO: NOT ALL P-FRAMES ARE EQUAL

A basic property of compressed video that has been exploited over the years is that different coded frames, and associated transmission packets, may have different importance, e.g. I-frames are more important that P-frames which are more important than B-frames, where importance in our context is in terms of the total mean-squared-error (MSE) distortion that is incurred if that frame is lost. This property of IPB frame coding, and also of scalable coding, is widely used to provide unequal (prioritized) treatment to the coded data and thereby provide improved performance. In particular, this property is exploited in R-D optimized streaming.

H.264 currently does not provide scalablity aside from the possible use of B-frames (temporal scalability). Furthermore, many H.264 applications do not use B-frames because of latency or memory constraints. To examine the use of SSS with non-scalable H.264, we consider when the video is coded with an initial I-frame followed by all P-frames, and no B frames. It is known that different P-frames have different importance, where, for example, the later P-frames in an MPEG Group of Pictures (GOP) typically are less important then the earlier P-frames in the GOP. However, a somewhat surprising observation is that P-frames can also differ in importance by a very significant amount. This is important since many applications primarily use P-frames, with very few I-frames, no B-frames, and no scalable coding. Therefore, by identifying and exploiting the varying importance of different P-frames in a sequence we can achieve improved performance. This approach extends trivially to include I and B frames.

All experiments are performed using JM 2.0 of the H.264/MPEG-4 AVC video compression standard. Four standard test sequences in QCIF format are used, *Carphone, Foreman, Mother & Daughter (MthrDhter),* and *Salesman*. Each is coded at a constant quantization level for an average PSNR of about 36 dB, at 30 fps, and has at least 350 frames. The first frame of each sequence is intra-coded, followed by all P-frames. Every 4 frames a slice is intra updated to improve error-resilience by reducing error propagation (as recommended in JM 2.0), corresponding to an intra update period of $N = 4 \times 9 = 36$ frames. Every P-frame fits within a single 1500 byte packet, hence in these experiments the loss of one packet corresponds to the loss of one P-frame. Every lost frame is replaced by the last correctly received frame, and distortion is measured after decoder error concealment. We assume that the initial I-frame is always correctly received to simplify the analysis.

The importance of each P-frame in each of the four sequences is shown in Figure 1. The total distortion that results for losing only a single isolated frame k is plotted, as the lost frame k varies from 2 to 300. The total distortion is the distortion that afflicts frame k and all subsequent frames from error propagation. It is clear from the plots that there is a significant amount of variability in the distortion that arises. The cumulative distribution function (CDF) of the distortion is also plotted for each sequence. The CDF identifies that there is a long tail, i.e. a sizable number of frames which produce considerably more distortion than the average frame. This observation is examined in more detail in Table 1, where various statistics are computed for each sequence. The median distortion is chosen to represent the distortion that is incurred for the loss of a typical P-frame. A very interesting observation is the relationship between the maximum and minimum distortions to the typical (median) distortion for each sequence. The importance of the various P-frames in a sequence varies by *two orders of magnitude*, where some packets lead to an order of magnitude more distortion than the median packet, and the median packet leads to an order of magnitude more distortion than the least important packets. This two-orders of magnitude difference in the max-to-min total distortions for P-frames within a sequence signifies that considerable gain can be achieved by identifying and exploiting the unequal importance of different P-frames and their associated packets.

3.1. R-D Optimized Processing of P-frames

The great diversity in importance of different P-frames can be exploited by placing the R-D information for each P-frame into

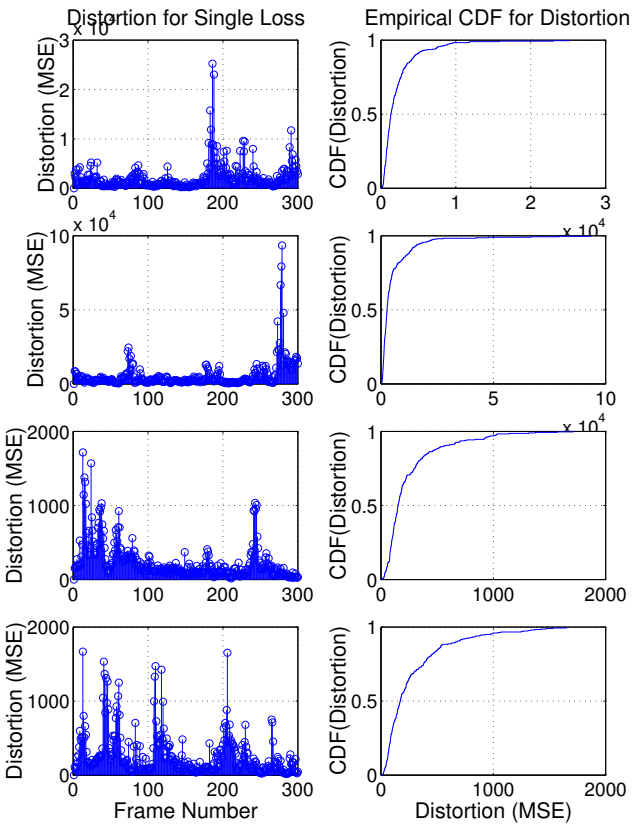

Fig. 1. Total distortion that results for losing any single P-frame, as a function of the lost frame (left) and corresponding CDF (right) for (top to bottom) Carphone, Foreman, MthrDhtr, and Salesman.

its packet header [1, 2], or in a R-D Hint Track (RDHT) associated with the coded video (and the additional storage available may support more information and sophisticated R-D optimization techniques [6]).

The prior discussion focused on the total distortion produced by a single lost/dropped packet, including the effects of all error propagation. The total distortion produced by simultaneously dropping multiple packets may be approximated as the sum of the distortions that result for losing each packet alone. This additive model for distortion ignores the important interdependencies that result from the loss of multiple packets within an intra-refresh period (e.g. a burst loss generally leads to more distortion than an equal number of isolated losses [8]). However, the additive distortion model is a reasonable first-order approximation (note that this additive model explicitly accounts for the great diversity in importance of different P-frames, in contrast to some prior additive models which assume a homogeneous model of the video). This approach also requires significantly less R-D information, which is an important consideration when placing R-D hints in each packet (but of lesser importance for R-D hints stored on a server). Furthermore, this simple model of additive distortion is generally applicable for a variety of different types of coders and types of media.

Assuming the additive model for total distortion, the optimal method for selecting among packets can be determined by associating for every packet j a corresponding utility measured in terms of distortion per bit, defined as $\lambda_j = D(j)/R(j)$. For example, if the total available bit rate is less than the video rate, a decision must be made as to which packets to transmit and which to drop, in order to minimize the total distortion while simultaneously satisfying the rate constraint. This problem is straightforwardedly solved by rank ordering the packets based on their utility and transmitting those with higher utility while dropping those with lower utility.

Sequence	Carphone	Foreman	MthDhtr	Salesman
Mean	2.25	5.62	0.25	0.28
Median	1.34	2.75	0.15	0.17
Max	25.24	93.46	1.71	1.67
Min	0.216	0.218	0.013	0.016
$\frac{Max}{Median}$	18.79	34.01	11.27	9.69
$\frac{Median}{Min}$	6.22	12.56	11.08	10.34
$\frac{Max}{Min}$	116.79	427.20	124.82	100.23

Table 1. Total distortion for losing a single packet (single P-frame). Some P-frame packets are worth an order of magnitude more than the typical (median) packet in terms of the total distortion incurred if they are lost, and some are an order of magnitude less important — thus some P-frames are *two orders of magnitude* more important than other P-frames within the same sequence.

4. SECURE ADAPTIVE STREAMING USING A SECURE-MEDIA RATE-DISTORTION HINT TRACK

This section proposes a technique based on a Secure-Media Rate-Distortion Hint Track (SM-RDHT), for designing and operating media streaming systems that can perform R-D optimized streaming with low complexity and while preserving the content security.

As background, the popular MPEG-4 File Format (MP4) incorporates a "hint track" which contains information about media type, packet framing, and timing information. This MP4 hint track provides "hints" to the streaming system that greatly simplifies the streaming. This is because the streamer no longer needs to (1) understand the compressed media syntax, and (2) analyze the media data in real time for packet framing and timing information.

The Secure-Media R-D Hint Track provides two important extensions of conventional MP4 hint tracks: (1) the R-D attributes for the media are derived and summarized in the hint track to enable low-complexity R-D optimized streaming [6], and (2) the R-D hint track is stored unencrypted (similar to the unencrypted headers in the secure scalable packets) while the media itself is encrypted. The SM-RDHT therefore enables a sender to read the unencrypted R-D hints and perform low-complexity R-D optimized streaming of the media – without having knowledge of the actual media.

The performance of a SM-RDHT system is shown in Figure 2, where the transmitted bit rate for the Foreman sequence is reduced below the original coded bit rate. The conventional system does not distinguish between P frames, and therefore randomly selects packets to drop, while the SM-RDHT system intelligently determines which packets to drop to maximize the quality while meeting the available bandwidth constraint. Note that non-scalable H.264 provides limited ability to scale the bit rate as compared to a scalable coder, however it is also clear that the SM-RDHT system provides dramatic improvements in quality over a conventional system when scaling is necessary.

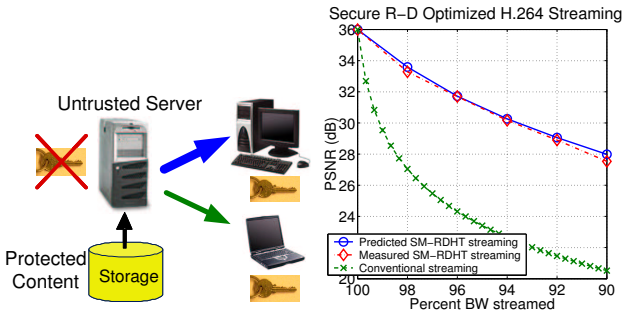

Fig. 2. Goal: Enable an untrusted sender to securely stream and adapt encrypted content for the available bandwidth without knowning what the content is. Performance of secure adaptive streaming using SM-RDHT versus conventional approach.

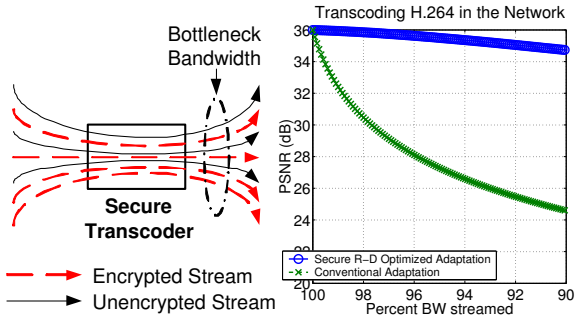

Fig. 3. Secure R-D optimized mid-network adaptation: a mid-network transcoder can securely adapt across a large number of streams to satisfy an output bandwidth constraint; the diversity gain from processing across many streams is evident.

5. SECURE TRANSCODING AT A MID-NETWORK NODE USING SECURE SCALABLE PACKETS

In this example, non-scalable H.264 video is packetized into secure scalable packets with unencrypted packet headers that provide R-D information about the importance of each packet. This information provides hints for the downstream transcoders. Preliminary investigation suggests that one byte of information in the unencrypted packet header may be sufficient, however this depends on the specific capabilities required. Mid-network transcoders read the unencrypted headers of each packet and select or discard each packet based on its importance and the network constraints. An important attribute of this approach is that the mid-network transcoder can perform R-D optimized adaptation across multiple packets of a single stream or across packets of multiple different streams.

The performance of this system is illustrated in Figure 3, where a large number of streams simultaneously pass through a node with limited output bandwidth, requiring the node to transcode the streams. Some of the non-scalable H.264 video streams are encrypted and some are not (but every packet includes transcoding hints) in order to highlight that both encrypted and unencrypted streams can be processed using the same low-complexity R-D optimization techniques.

To illustrate the potential benefits of diversity gain from transcoding across multiple streams we consider the following somewhat artificial experiment which nevertheless highlights the key points. This plot shows an estimated upper bound on performance (estimated using the additive model and the derived R-D information), where all four test sequences are simulated to be streamed at all possible phases relative to each other. In this manner we simulate a much larger number of test sequences than are available, or alternatively this is equivalent to streaming the four sequences and the transcoder examining the entire 10 sec length of each sequence to determine which packets to drop. The transcoder examines all packets from all sequences within a time window and selects which packets to transmit and which to discard based on the importance of each packet and the output bandwidth constraint. This is achieved by reading the unencrypted header of each packet in the window, sorting the results (a partial sort is sufficient), and selecting/discarding based on the priority.

The performance in Figure 3 is better than that in Figure 2 as is evident by the smaller drop in PSNR. This is due to the diversity gain from processing across streams. Specifically, by transcoding across streams we can exploit the fact that many frames of MthrDthr and Salesman are of lesser importance than those of Foreman, and hence can be preferentially dropped. Therefore, the secure transcoder is attempting to maximize the quality (minimize the total distortion) across all of the streams.

6. SUMMARY

This paper examined the problem of how to efficiently stream and adapt encrypted non-scalable media, while preserving end-to-end security. We showed that the Secure Scalable Streaming framework could be applied to non-scalable H.264 coded video, by identifying and intelligently distinguishing the importance of different P-frames. Specifically, we identified over two orders of magnitude difference in the importance of P-frames within a single sequence, and an even large difference can exist between multiple sequences. We examined two techniques where the media is encrypted and associated R-D information is placed in unencrypted packet headers or in the Secure Media RDHT enabling efficient R-D optimized streaming and adaptation at the sender, or at a mid-network node or proxy, for non-scalable H.264 video. Furthermore, this approach for R-D optimized streaming and adaptation is useful for streaming both encrypted and unencrypted content.

7. REFERENCES

[1] S.J. Wee and J.G. Apostolopoulos, "Secure scalable video streaming for wireless networks," *IEEE ICASSP*, May 2001.

[2] S.J. Wee and J.G. Apostolopoulos, "Secure scalable streaming enabling transcoding without decryption," *IEEE ICIP*, Oct. 2001.

[3] S.J. Wee and J.G. Apostolopoulos, "Secure scalable streaming and secure transcoding with JPEG-2000," *IEEE ICIP*, Sept 2003.

[4] *ISO/IEC JPEG-2000 Security (JPEC) Committee Draft*, April 2004.

[5] Z. Zhang, Q. Sun, G. Qiu, Y.Q. Shi, and Z. Ni, "A unified authentication framework for JPEG2000," *IEEE ICME*, 2004.

[6] J. Chakareski, J. Apostolopoulos, S. Wee, W. Tan, and B. Girod, "R-D hint tracks for low complexity R-D optimized video streaming," *IEEE ICME*, June 2004.

[7] C. Venkatramani, P. Westerink, O. Verscheure, and P. Frossard, "Secure media for adaptive streaming," *ACM Multimedia*, pp. 307–310, Nov 2003.

[8] Y. Liang, J. Apostolopoulos, and B. Girod, "Analysis of packet loss for compressed video: Does burst-length matter?," *IEEE ICASSP*, April 2003.