Invited paper in "Network-Aware Multimedia Processing and Communications" special session at IEEE ICIP 2006.

ARCHITECTURAL PRINCIPLES FOR SECURE STREAMING & SECURE ADAPTATION IN THE DEVELOPING SCALABLE VIDEO CODING (SVC) STANDARD

John G. Apostolopoulos

Streaming Media Systems Group, Hewlett-Packard Labs, Palo Alto, CA, U.S.A.

ABSTRACT

Scalable video coding has long been known to provide important functionalities such as low-complexity adaptation for diverse clients with different resources and for delivery over heterogeneous networks with time-varying available bandwidths. Recent advancements in scalable video coding have significantly improved the achievable compression performance, and the Scalable Video Coding (SVC) standard is currently under intense development. An additional capability of scalable coding, which we believe has received less attention than it deserves, is the possibility with careful cross-layer design to support secure adaptive streaming at an untrustworthy sender and secure adaptation at an untrustworthy mid-network node. Building on the Secure Scalable Streaming framework for video, and its realization within the JPEG-2000 Security (JPSEC) standard, we describe the architectural principles and design details necessary so that SVC can also enable secure adaptive streaming at a sender and secure R-D optimized adaptation at an untrustworthy mid-network node.

Index Terms— Secure streaming, secure adaptation, scalable video coding, SVC, secure transcoding, secure scalable streaming

1. INTRODUCTION

Scalable video coding provides many valuable capabilities, including the ability to adapt the coded media for delivery over networks with different or time-varying available bandwidth and/or to diverse clients with different resources such as display sizes, etc. An additional capability which may be provided by scalable coding, and which we believe may be receiving less attention than it deserves, is the ability to perform *secure* adaptive streaming at a sender and *secure* adaptation (transcoding) at a mid-network node. We believe these functionalities may be practically very useful in the future.

Recent advancements in scalable video coding have led to significant improvements in compression [1], and the Scalable Video Coding (SVC) standardization effort is currently under intense development [2, 3]. SVC is expected to provide excellent compression and a rich set of scalabilities (including spatial, temporal, and quality) and freedom to adapt across these dimensions. These valuable scalability capabilities can be straightforwardly used in many practical applications — where security is not required. However, by incorporating security considerations within the SVC design, it is possible to preserve and provide SVC's scalability capabilities even in secure settings. These settings are identified in Section 2.

Security and flexible handling of media, including adaptation, are traditionally conflicting goals, and security is often incorporated in a media distribution system in a media-unaware manner. For example, the media is typically encrypted as a block of data and then stored as an encrypted file or delivered using a reliable delivery mechanism. This approach protects the media, but also prevents many valuable capabilities. Important recent techniques such as Secure RTP [4] combine media-aware application-level framing with security, but they do not address the challenges of secure adaptation.

This paper examines how SVC can be designed so that media-aware protection can be used to simultaneously achieve end-to-end security and flexible secure adaptive streaming and secure mid-network adaptation (transcoding) of the protected content. We focus on the security services of confidentiality and authentication, and describe the techniques that make this possible. This paper draws from the Secure Scalable Streaming (SSS) framework which provided end-to-end security and mid-network secure transcoding for scalably coded video [5, 6], and the recent JPEG-2000 Security (JPSEC) standard which is the first standard to apply these techniques to media [7].

For a review of the current status of the rapidly evolving SVC standard see [2, 3]. Note that the proposed functionalities are possible to achieve by abstracting various features of SVC. For example, the proposed techniques do not depend on the details of how intra-coding is performed, or the context of the arithmetic coder, or specific details of the scalable coding. While these issues are critical for a successful SVC, they do not directly effect this work. Our approach draws parallels to application-layer framing which uses media-aware transport to achieve improved error resilience to packet losses, for which H.264/MPEG-4 AVC developed the network abstraction layer (NAL) and associated NAL units. The proposed design may be thought of as a security abstraction layer (SAL).

This paper continues by describing the desired SVC functionalities in a secure setting. Section 3 describes the framework from SSS and JPSEC that is applicable to provide the desired SVC functionalities, including the concepts of progressive encryption, secure scalable packets, and secure R-D optimized adaptation. Sections 4 and 5 discuss encryption and authentication for SVC, respectively.

2. DESIRED MEDIA-SECURITY FUNCTIONALITIES

This section highlights some basic functionalities and attributes that are desirable for SVC to provide in a secure context.

Secure Streaming: The primary goal of secure streaming is to protect the media content from eavesdroppers, thereby necessitating the use of end-to-end encryption where the media is encrypted at the sender and decrypted only at the receiver. It can also be beneficial to have creation-to-consumption security, where the content is encrypted by the content creator and decrypted at the content consumer, everywhere in between the content is kept in encrypted form.

Secure adaptation (transcoding) at untrustworthy mid-network nodes: It is often beneficial for a mid-network node or proxy to be able to adapt content that it receives to match downstream dynamic network conditions or receiving clients. However, when the content is encrypted the conventional approach is to decrypt the stream, adapt, and then re-encrypt. This is not an acceptable solution as it breaches the end-to-end security and leads to many vulnerabilities. In addition to these vulnerabilities, in many situations it is desirable to perform adaptation at mid-network nodes or proxies that are untrustworthy, and therefore should not have access to the key. There-

fore, the goal is to be able to adapt the encrypted content without requiring the key. This task of simultaneously achieving the conflicting goals of (1) adapting at intermediate, possibly untrusted network nodes, while (2) preserving end-to-end security, seemingly leads to a paradox because intuitively to adapt in the middle of the network you want to know what the bits are, but the goal of end-to-end security is to prevent any intermediate node from knowing what the bits are. This problem can be overcome by co-designing the coding, security, and packetization [5, 6].

Secure adaptive streaming at untrustworthy sender: In addition to securely adapting the content at a mid-network node, another important and related capability is to enable a (potentially untrustworthy) sender to stream and adapt the streaming of encrypted content without knowing what the content is. For example, content creators typically prefer to protect the content themselves, and would like the media distributors to appropriately distribute the content without unprotecting it. Similarly, the media distributors would also prefer if possible to be able to adapt the delivery of the content without requiring access to the keys or unprotecting it as then they are not liable for any breaches.

Creation-to-consumption security: In addition to the notion of end-to-end security (from sender to receiver) that is common in the Internet space (e.g., VPNs, IPsec, SSL), as mentioned above for the untrustworthy sender there is the important capability of protecting the content at the content creator and unprotecting it only at the consumer – creation-to-consumption security – where everywhere in between (including untrustworthy streaming servers and mid-network nodes) the content is kept in protected form. Creation-to-consumption security is common for file-based delivery, however once again our focus is on adaptive streaming.

For simplicity the above discussion has focused on providing confidentiality (via encryption), however a variety of security services are desired, e.g., authentication, access control (see, e.g. [8, 7]).

3. FRAMEWORK FOR PROVIDING FUNCTIONALITIES

The above media-security functionalities can be provided within a framework referred to as Secure Scalable Streaming (SSS) which involves the careful co-design of the compression, security, and packetization. This approach allows for the creator-to-consumer delivery of encrypted media content while enabling mid-network adaptation to be performed without decryption. This capability is referred to as *Secure Adaptation* (*Secure Transcoding*) in order to stress that the adaptation is performed without requiring decryption (without requiring the key) and therefore preserving the end-to-end security. Note that with this approach, the media content is protected throughout the delivery chain from the content creator to each receiving client, so the encryption keys are only available to these entities, and not to the streaming sender which performs the secure adaptive streaming, or to the mid-network node which performs the secure adaptation. SSS was designed for scalably coded media, however it is also applicable, though to a much more limited extant, to non-scalably coded media such as H.264/AVC video [9].

The basic idea of SSS is to co-design the coding, encryption, and packetization to facilitate intelligent discarding of data, where after discarding the remaining data can still be decrypted, authenticated, and decoded. Furthermore, the creation of "hints", such as rate-distortion (R-D) hints, which can be used to intelligently direct the adaptation. For a streaming server, the hints can be placed as unencrypted hints in the Secure R-D Hint Tracks within a modified MPEG-4 File Format that is stored at the server, thereby enabling secure R-D optimized adaptive streaming at an untrustworthy sender.

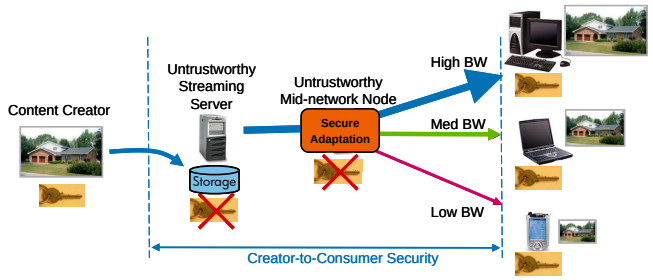

Fig. 1. Secure adaptive streaming & secure mid-network adaptation.

Similarly, hints can also placed in unencrypted packet headers of Secure Scalable Packets (described below) to enable R-D optimized secure adaptation at untrustworthy mid-network nodes.

Note that while the use of R-D hints at a server and mid-network node are conceptually similar, their operation is quite different. A server typically has ample room to store large amounts of extra data without penalty, while the overhead from packet headers directly reduce the delivered data. Furthermore, while the server has access to the entire media, a mid-network node only has access to a small window of packets within which it must perform its processing.

SSS can provide secure, coarse-grain to fine-grained (layer to packet to sub-packet) adaptation capabilities. Furthermore, fine-grained secure adaptation can be performed in a R-D optimized manner by using the information contained in the unencrypted packet headers. We next examine the granularity that can be achieved with SVC and how to perform R-D optimized streaming and adaptation.

3.1. Granularity of Secure Adaptation: Layer vs. Packet vs. Truncated Packet

This section examines the important issue of the granularity that can be achieved with secure adaptive streaming and secure mid-network adaptation. The level of granularity may be one of the following:

- Sequence or group of pictures (GOP)
- Scalable layer
- Packet
- Sub-packet (e.g., truncated packet)

Sequence and GOP level granularity are available in both scalable and non-scalably coded video. In addition, SVC creates multiple layers of compressed data that are prioritized in terms of importance, e.g., spatial, temporal, or quality scalability. These enhancement layers can be independently encrypted and sent in separate packet flows, allowing a mid-network node to discard a flow containing low-priority data while forwarding the flows containing high-priority data which are then received, decrypted and decoded at the receiver. This can also be straightforwardly achieved in a single flow, when each packet's header identifies its layer information. This approach provides a fine solution in certain contexts. However, the granularity of the above adaptation is limited by the granularity of the layers, which is not sufficient in certain contexts.

Packet networks operate with a packet granularity, so the operation of packet select/discard is natural in this context. However, sometimes even finer granularity can be beneficial. For example, in streaming applications a mid-network node typically has access to only a small number of packets to perform its adaptation.

The ability to have sub-packet granularity, such as by performing packet truncation, provides much finer granularity and improved

performance as compared to operations such as packet select/discard or the much coarser select/discard of scalable layers. For example, consider the case when a transcoder has access to only two packets and must reduce the bit rate by 10 %. Clearly the ability to truncate a packet could lead to better performance than being forced to discard a packet. Furthermore, when only one packet is available and a reduction in bit rate of 10 % is required then the inefficiency is even more apparent — if limited to packet select/discard then no data will be delivered. These simple examples illustrate that the capability to truncate packets can be quite valuable.

The above processing can be achieved by first creating *scalable packets*, by deliberately placing the scalable coded data into packets in a prioritized manner so that adaptation can be performed via a packet truncation (or discard) operation. The scalable packet payload data is then encrypted using a *progressive encryption* method to form *secure scalable packet* data. Progressive encryption methods have the property that after truncation the truncated encrypted data can still be decrypted and decoded. Note that progressive encryption methods correspond to conventional encryption methods — which are used in a non-conventional manner where portions of the encrypted data are discarded. Also note that while the discussion focuses on the operation of packet truncation because of its conceptual and practical simplicity, one can discard the head of the packet payload or arbitrary ranges of bytes in the interior of the packet payload, where the preferred form of discarding of data depends on a variety of issues which are not discussed here because of limited space.

Importantly, SSS packets include an *unencrypted header* which contains information that is used to direct the subsequent SSS adaptation. This information may include a relative or absolute measure of each packet's importance, a series of recommended truncation points for each packet, or hints to guide downstream nodes to perform adaptation operations such as resolution reduction or rate-distortion optimized bit rate reduction.

Secure adaptation is performed by reading the packet header and appropriately discarding or truncating each packet to meet, e.g., the downstream bit rate constraint. This operation is referred to as *secure adaptation* (or *secure transcoding*) to emphasize that it does not require decryption and therefore preserves the end-to-end security. The key idea once again is that adaptation is performed by an intelligent discarding of data, without requiring knowledge of what the data actually is [5, 6].

R-D optimized secure adaptation is possible because the necessary information to perform the R-D optimized processing can typically be distilled into a small amount of data, and made available with the unencrypted hints (e.g., unencrypted hint tracks at the streaming server or unencrypted packet headers). For example, assuming the total distortion and bit rate are additive across packets, we can minimize the total distortion subject to a bit rate constraint by truncating packets so that they operate at the same slope on their respective R-D curves, such that the sum of the corresponding bit rates satisfies the rate constraint. For instance, Figure 2 shows an example where a mid-network node receives two packets and has a transmission bitrate constraint where the output rate must be less than 3/4 of the input rate. A conventional node in the network would meet this bit rate constraint by discarding one of the two packets. However, with the unencrypted packet headers which contain R-D information for the two packets, the transcoder can estimate the R-D curves for each packet and then truncate each packet so that it is operating at the same slope on the R-D curve for each packet while satisfying the bit rate constraint. Intuitively, one way to achieve this is by truncating, in parallel, across the packets based on the slopes of their R-D curves, until the desired rate reduction is achieved. The resulting truncated secure scalable packets can be decrypted and decoded by the receiver with a reconstructed quality that depends on the received data [5, 6].

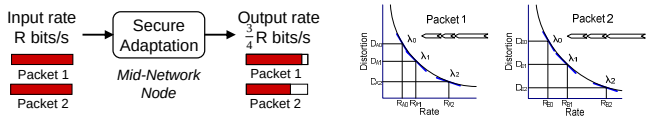

Fig. 2. R-D optimized secure adaptation across two packets is achieved by appropriately truncating each packet so that they operate at the same slope λ on their R-D curves.

An important tradeoff exists between packet header size and secure adaptation performance. The more information contained in the packet header, the more accurate the estimate of the packet's R-D curves, and the better performance. However, the larger the header the greater the overhead. Careful header design is necessary, and depends on, e.g., whether the encrypted packet payload is fully embedded with byte granularity or whether it has coarser granularity. There is also an issue of potential leakage of information, since the unencrypted header describes some attributes of the coded media.

Secure scalable packets provide a number of properties: they are scalable to enable downstream adaptation by operations such as packet truncation or discarding, encrypted to provide end-to-end security, provide hints to optimally guide the adaptation, and independently decodable to provide resilience to packet losses.

SVC recently introduced a new SVC-specific NAL unit type, which provides bitstream scalability at level of NAL packets (packet select/discard). Furthermore, with FGS it supports truncation of enhancement layer packets at arbitrary points. Therefore, SVC has basic support for packet and sub-packet level processing and the ability to include certain header information describing packet payloads. This enables scalable packets and adaptation of scalable packets as described in SSS. With further consideration of security issues SVC may support secure scalable packets and R-D optimized secure adaptive streaming and secure mid-network adaptation.

4. COMMENTS ON ENCRYPTION AND SVC

A straightforward approach to encrypt SVC coded video is to use media-aware packetization and packet-by-packet encryption, based on application-level framing principles. Specifically, packets are designed to be independently decryptable, authenticatable, and decodable, so that even if packet loss occurs the receiver can decrypt, authenticate, and decode all of the received packets. Security of RTP flows is practically very important, and therefore an extension of RTP, referred to as Secure RTP (SRTP), was developed which provides confidentiality, message authentication, and replay protection for the RTP traffic, as well as for its associated control traffic RTCP [4]. SRTP, and similar approaches (e.g., [10]), provide the basic security services required for secure streaming between a sender and a receiver. However, in addition to providing secure streaming, SSS provides the ability to securely adapt the protected content.

A very important design rule, which we strongly recommend to follow, is to design SSS-based systems using existing, highly studied, cryptographic primitives, as opposed to designing new primitives, since any new primitives are likely to have subtle, hidden flaws. In addition, there is no need to create new cryptographic primitives. The innovation with SSS is in using well-known and well-studied cryptographic primitives in a different manner from how they are

conventionally used. For example, SSS can be used with a variety of standard encryption methods, including block ciphers such as the Advanced Encryption Standard (AES), and stream ciphers such as RC4, or stream ciphers created out of block ciphers (e.g., AES using OFB or CTR). Similarly, authentication may be provided using a number of popular and well-studied authentication primitives.

An example of the use of Secure Scalable Packets is shown in Figure 3 where the Bus sequence (QCIF, 150 frames, 30 fps) is coded using the SVC reference software [2, 3] into a H.264 base layer (223 kb/s) and one FGS layer (total rate 466 kb/s). We examine the performance with/without the use of (1) progressive encryption of the packets using AES in CTR mode which enables decryption of truncated packets, and (2) unencrypted packet headers which guide the adaptation of each individual packet. The "Discard All" curve is a crude, lower bound estimate of the performance if conventional IPSEC-type packet encryption is used. When progressive encryption is used, but without the unencrypted headers for guidance, the performance is estimated by truncating all packets equally to meet the rate constraint. The "Heuristic" and "R-D Optimized Truncate" curves are illustrative examples of what can be achieved by secure adaptation when using both progressive encryption and unencrypted R-D packet headers.

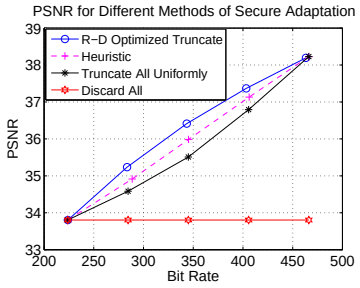

Fig. 3. Example performance of secure adaptation without/with (1) progressive encryption, (2) unencrypted R-D packet headers.

5. COMMENTS ON AUTHENTICATION AND SVC

In this paper, authentication refers to verifying the source and integrity of the data, i.e., whether or not the data has been accidentally or maliciously altered. Conventional "data" authentication approaches employ message authentication codes (MACs) or digital signatures and assume reliable delivery, i.e., all of the original signed data is available at the receiver to perform the verification. Clearly this is not the case for video streaming over UDP where packet losses may occur — or when we would like to use SVC's flexibility to purposefully adapt the content at the sender or at a mid-network node.

To clarify this important point, for authenticated video only the video packets which are both received and authenticated are decoded and contribute to the reconstructed video — video data which are received but are not authenticated are equivalent to being lost. Therefore authenticated video has two penalties as compared to conventional video. The first is the additional rate (overhead) required for the authentication information such as MAC(s) or digital signature(s). This part of the penalty can be reduced by amortizing the overhead cost over a group of many packets. However, there is an important tradeoff between verification probability and overhead, because if any one of the packets in the group is lost then the group can not be verified. The second penalty corresponds to the amount of data which is received but unauthenticatable, and hence useless. The first cost is deterministic, and known to the content creator, however the second cost depends on the channel losses and is unknown to the content creator and can only be estimated. A Lagrangian R-D framework can be used to determine the best authentication operation, assuming knowledge of the channel loss rate, to optimally balance rate overhead versus expected distortion from received but unauthenticatable video packets [11].

SVC authentication should therefore be designed to be resilient to packet losses, and not to adversely hinder secure adaptive streaming and secure mid-network adaptation. Furthermore, since our goal is to adapt the SVC content, it is important to be able to distinguish between allowed versus malicious adaptations. Specifically, the receiver must be able to verify that any alterations were performed in a valid and permissible manner. This includes authentication when discarding scalable layers, packets, and packet truncations.

A design principle to efficiently achieve the above, is to align the authentication dependencies with the SVC coding dependencies. This will ensure that secure adaptation along pre-defined paths (given by the dependencies between scalable layers) will not negatively affect the verification probability of the remaining delivered SVC data. This can be achieved via Merkle hash trees, which can easily be non-binary and unbalanced. Additional adaptations at the sender or mid-network node may be supported by adding "patches" based on the specific adaptation performed, which preserve the receiver's ability to verify the received data.

6. SUMMARY AND ON-GOING WORK

This paper examined how secure adaptive streaming at a sender and secure adaptation at a mid-network node can be supported in the developing SVC standard. This is achievable by leveraging the SSS framework, as adopted in the JPSEC standard. We believe that these functionalities may be practically quite useful in the future. We are continuing our study of applying SSS to the developing SVC standard, and if the above functionalities are deemed useful and within the scope of the standardization effort, we will contribute our recommendations to the SVC effort.

7. REFERENCES

[1] J.-R. Ohm, "Advances in scalable video coding," *Proceedings of the IEEE*, January 2005.

[2] J. Reichel, H. Schwarz, and M. Wien, *Scalable Video Coding - Joint Draft 4, JVT-Q201*, ISO/IEC and ITU-T, October 2005.

[3] J. Reichel, H. Schwarz, and M. Wien, *Joint Scalable Video Model JSVM-4, JVT-Q202*, ISO/IEC and ITU-T, October 2005.

[4] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "Secure real-time transport protocol (SRTP)," *IETF RFC 3711*, 2004.

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

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

[7] *ISO/IEC JPEG-2000 Security (JPSEC) Final Committee Draft*, Nov. 2004.

[8] Y. Wu, D. Ma, and R. Deng, "Progressive protection of JPEG2000 codestreams," *IEEE ICIP*, October 2004.

[9] J.G. Apostolopoulos, "Secure media streaming & secure adaptation for non-scalable video," *IEEE ICIP*, October 2004.

[10] Internet Streaming Media Alliance (ISMA), *ISMA Implementation Specification: Encryption and Authentication Specification*, Feb, 2004.

[11] Z. Zhishou, Q. Sun, W. Wong, J. Apostolopoulos, and S. Wee, "Rate-distortion optimized streaming of authenticated video," *ICIP*, 2006.