Compressed-Domain Video Processing for Adaptation, Encryption, and Authentication

Razib Iqbal, *Student Member, IEEE,* Shervin Shirmohammadi, *Senior Member, IEEE,* Abdulmotaleb El Saddik, *Senior Member, IEEE,* and Jiying Zhao, *Member, IEEE*

***Abstract*— Availability of sophisticated handheld devices has made possible different multimedia applications to run on the go. Nevertheless, the limitations of the screen resolution and the processing capability of these devices prevent us from representing multimedia information in the same way as is done for regular displays. Simultaneously, with the advances in wireless communication, multimedia content transmissions now have become possible through the 3G and beyond 3G networks. However, the communication capabilities of different devices differ due to the limited bandwidth which varies because of either signal fading or network unavailability. In consequence, the importance of adapting the media according to users' context sets the primary challenge towards seamless access in a ubiquitous computing environment. Conventional video processing approaches entail cascaded operations of decompression, processing and recompression within an adaptation scenario. In this paper, we present a framework based on MPEG-21 gBSD that provides adaptation, encryption and authentication of video, specifically H.264 bitstream, in the compressed domain. The proposed scheme does not necessitate any cascaded compression and decompression, and has demonstrated faster performance in experimental results compared to cascaded approach. A complete 3-in-1 system architecture is detailed along with the experimental evaluations of the system supporting the design concept.**

***Index Terms*—Multimedia communication, video adaptation, video authentication, video encryption, H.264, MPEG-21, gBSD**

I. Introduction

UBIQUITOUS computing concept permits end users to have access to multimedia and digital content anywhere, anytime and in anyway they want. Today's multimedia consumers expect to access content using any platform, from PC or laptop to cell phone, PDA, Sony PSP, iPod, and many other new and upcoming networked-computing devices, and over various kinds of networks be it high speed DSL, wireless LAN, or others. Indeed the driving force behind Universal Multimedia Access (UMA) demands a generic adaptation procedure to access multimedia contents seamlessly in both wireless and wired networks. Since the diversity of devices via which multimedia content are accessed and interacted with has grown significantly, and continues to do so, it is infeasible to adapt the playback environment to accommodate all various media formats, which continue to grow. The efficient solution is to adapt the media content for the playback environment used by the consumer and providing them access to a large variety of items in an interoperable manner. Simultaneously, security and digital right management (DRM) of multimedia data has become an emerging concern, especially in UMA networks. Therefore, encryption and authentication operations should be taken care of not only for serving sensitive digital content, but also for offering security and integrity as an embedded feature of the adaptation practice. A straightforward solution to adaptation, encryption and authentication can be achieved by allocating some intermediary nodes in between the content provider and the consumer to perform manifold decoding and encoding operations such as decompression, adaptation, encryption, authentication, and recompression. While simple, this approach requires a large amount of processing time or processing power on the intermediary nodes, not to mention absolute trust in them, which can be compromised by a third party. An alternative solution would be to produce and store content in several formats taking into account a

The authors are with the School of Information Technology and Engineering, University of Ottawa, 800 King Edward Ave., Ottawa, ON, Canada, K1N 6N5 (phone: 1 613 562 5800 ext. 6206, e-mail: {riqbal|shervin}@discover.uottawa.ca; {elsaddik|jyzhao}@site.uottawa.ca).

wide variety of possible user devices and preferences, making an appropriate selection at the delivery time. However, new consumer devices with different capabilities and network access are emerging on a daily basis and it is impractical to pre-store all media for all possible contexts. Another approach is to perform partial selection of streams from a multiple pre-encoded bitstream. However, for real time applications, it could be impractical to apply the above solutions because of the processing time, availability of resources, and limited space. A real-time and efficient solution to perform the necessary operations of adaptation, encryption, and authentication in the compressed domain, without decompression and recompression or without storing the media in various formats, will be a significant expansion.

In this paper, we present an integrated system for compressed-domain video processing to achieve the above goal. We propose a 3-in-1 system by incorporating our recent research in compressed domain video authentication to our previous works in compressed video adaptation [1] and encryption [2]. Our architecture utilizes MPEG-21 generic Bitstream Syntax Description (gBSD) to perform the necessary modifications in the compressed bitstream, hence it is codec-independent in that intermediate nodes performing the operations need not have knowledge of the specific codec in use. To the best of our knowledge, no other work presents an MPEG-21 based architecture that performs adaptation, encryption, and authentication of H.264 video, entirely in the compressed domain. The architecture is portrayed in the form of a proof of concept prototype where temporal adaptation of H.264 video has been performed for both pre-coded and live video streams. We also show that exploiting the gBSD, a perceptual encryption scheme, and a hard authentication technique can be integrated in the adaptation system for encryption and authentication of the video content respectively, and demonstrate this feasibility with performance evaluation of the prototype and its comparison with a conventional cascaded system. The rest of the paper is organized as follows: in section II, the motivation for utilizing MPEG-21 gBSD and H.264 video format is discussed. A brief review of the related work is detailed in Section III. The proposed architecture for video adaptation procedure for both pre-coded and live video streaming is depicted in Section IV. Section V extends the application of Digital Item (DI) towards authentication and encryption of the adapted video data. A discussion on our work is presented in section VI. Finally, the paper ends with some insight of what can be extended in the future in Section VII.

II. MOTIVATION FOR APPLYING MPEG-21 GBSD AND H.264

Without a standard metadata support, variety and incompatibility of adaptation approaches in a distributed multimedia environment makes adaptation procedures unlikely. ISO/IEC 21000 Multimedia Framework was initiated because of the lack of interoperability among advanced multimedia packaging and distribution applications. Part 7, Digital Item Adaptation (DIA), of the MPEG-21 framework [3] specifies the syntax and semantics of tools that may be used to assist adaptation of a Digital Item. A DI is denoted as a bitstream together with all its relevant descriptions. The bitstream can be audio, video, or any other media; in simplified terms, the DI is basically the media such as audio, video, etc plus its metadata description. The advantage of MPEG-21 DIA is that it will benefit all the stakeholders in the multimedia delivery and consumption chain (e.g. Service and Content Providers, Network Operators, Device Manufacturers and End Users) in terms of interoperability and compatibility of products. This standard can be used in any application domain because it has been designed in a protocol and application independent way. An overview of DIA, its use in multimedia applications, and report on some of the ongoing activities in MPEG on extending DIA for use in rights governed environments are available in the literature [4][5] and are beyond the objective of this paper.

The concept of MPEG-21 DIA and the structure of the DIA engine are illustrated in Figure 1. From this figure, we can see that, a DI is subject to a Resource Adaptation Engine and a Description Adaptation Engine, which together produce the adapted Digital Item. In the specification, DIA tools are clustered into several categories. For example, Bitstream Syntax Description (BSD) describes the syntax (the high-level structure) of a binary media resource such as video. With Bitstream Syntax Description Language (BSDL), which is an XML Schema based language and standardized in the MPEG-21 framework, it is possible to design specific Bitstream Syntax Schema (BS Schema) describing the syntax of a particular coding format. A normative processor named BSDtoBin is specified in MPEG-21 to generate the adapted bitstream for BSDL. Since, BSDL provides a way for describing bitstream syntax with a

codec specific BS Schema, an adaptation engine consequently requires knowing the specific schema. Therefore, a generic Bitstream Syntax Schema (gBS Schema) is specified in the MPEG-21 framework to offer a format independent adaptation procedure. The gBS Schema introduces the means to describe hierarchies of syntactical units and addressing means for efficient bitstream access. A description conforming to this schema is called a generic Bitstream Syntax Description (gBSD), which provides an abstract view on the structure of the bitstream that can be used in particular when the availability of a specific BS Schema is not ensured. The gBSD is essentially a metadata definition of the media, written in the form of XML. For example, for video, the gBSD represents each frame's number, starting position in the bitstream, frame type and length of each frame, among other information, as seen in Figure 2. The benefit of applying gBS Schema is mainly the Codec independence, Semantic marking of syntactical elements, and Hierarchical descriptions of a bitstream that allows grouping of bitstream elements for efficient, hierarchical adaptations. To generate adapted bitstream, the gBSDtoBin processor is normatively defined in the MPEG-21 framework for gBS Schema. In our architecture, we make extensive use of the gBSD for all of the compressed domain operations on the video.

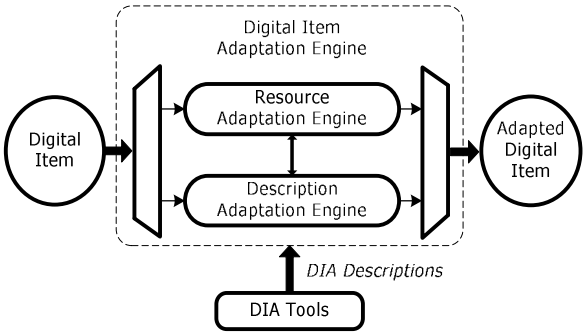

FIGURE. 1. MPEG-21 DIGITAL ITEM ADAPTATION

```
<?xml version="1.0" encoding="UTF-8" ?>
<dia:DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS" xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <dia:Description xsi:type="gBSDType">
    <Header>
      <ClassificationAlias alias="MV4" href="urn:mpeg:mpeg4:video:cs:syntacticalLabels"/>
      <DefaultValues addressUnit="byte" addressMode="Absolute" globalAddressInfo="test.raw"/>
    </Header>
                <gBSDUnit syntacticalLabel="NALUnit" start="0" length="23" />
                <gBSDUnit syntacticalLabel="Frame-1" start="23" length="6693" marker="IDR" />
                <gBSDUnit syntacticalLabel="Frame-2" start="6716" length="6177" marker="P" />
                <gBSDUnit syntacticalLabel="Frame-3" start="12893" length="6186" marker="P" />
                <gBSDUnit syntacticalLabel="Frame-4" start="19079" length="6177" marker="P" />
                <!-- ...and so on... -->
    </dia:Description>
</dia:DIA>
```

FIGURE. 2. SAMPLE GBSD REPRESENTATION

In the past decade, a fair number of video codecs like H.261, MPEG-2, and H.263 have evolved. H.264 is the latest video coding and compression standard by ITU-T and ISO/IEC as International Standard MPEG-4 part 10 Advanced Video Coding (AVC). The advanced compression technique, improved perceptual quality, network friendliness and versatility of the codec [6][7] drives it to outperform all the previous video coding standards. One of the major reasons behind choosing H.264 for our framework is its multiple reference pictures for motion

compensation. In H.264, slice sizes are flexible where a picture can be split into one or several slices. Slices are self-contained and can be decoded without using data from other slices. H.264 offers an entropy coding design which includes Context-Adaptive Binary Arithmetic Coding (CABAC) and Context Adaptive Variable Length Coding (CAVLC). Since in H.264 data is entropy coded, in order to achieve byte-alignment, sequence of bits is being padded by the encoder when necessary. Bitstream Data Unit (BDU) is defined as a unit of the compressed data which may be decoded independently of other information at the same hierarchical level. A BDU can be for example a frame or a slice of a frame, and is used in our approach for temporal adaptation and authentication of the adapted video respectively.

III. Related Work

There have been many research activities and advances in video adaptation, encryption and authentication in the past decade. Prominent research revealed during our study will be highlighted in this section. In [8], a comprehensive overview of digital video transcoding in terms of architecture, techniques, quality optimization, complexity reduction, and watermark insertion was presented. In [9], authors investigated quality adaptation algorithms for scalable encoded variable bit rate video over the internet to maximize perceptual video quality by minimizing quality variation. Bonuccellit et al. [10] introduced buffer-based strategies for temporal video transcoding adding a fixed transmission delay for buffer occupancy in frame skipping. A frame is skipped if the buffer occupancy is greater than some upper value, and it is always transcoded if the buffer occupancy is lower than some lower value, provided the first frame (i.e. the I-frame) is always transcoded. Group of Pictures (GOP) level rate adaptation scheme was introduced in [11] for a single stream, variable target bitrate H.264 encoder, which allows each group of pictures to be encoded at a specified bitrate, using a dynamically updated table to select the starting quantization parameter for each GOP. Block Adaptive Motion Vector Resampling (BAMVR) method was proposed in [12] to estimate motion vector for frame rate reduction in H.264. However, the transcoder follows straightforward cascading architecture of the decoder and encoder. Devillers et al. proposed a BSD based adaptation in streaming and constrained environments [13]. In their framework, the authors emphasized on BSD based adaptation applying BS Schema and BSDtoBin processors. In our case, we have used gBSD, which provides an abstract view on the structure of the bitstream that can be used in particular when the availability of a specific BS Schema is not ensured, and gBSDtoBin processor is being used for transformation.

The simplest way to encrypt video is perhaps to consider the whole stream as a 1-D array and then encrypt this 1-D stream with the encryption key. However, with advancement of time, video encryption mechanism considering sensitivity of digital video has been developed for layered video compression techniques. Partial or selective encryption algorithms such as [14] and [15] are deployed on selective layers. In this case, the content is encrypted and thus decrypted exploiting the inherent properties of the video coding layer(s). On the other hand, for video authentication, watermark can be embedded in the uncompressed domain, during the compression process or after the compression process. The DCT coefficient based embedding systems [16]-[18] embed binary watermark bits in the DCT domain derived from different extracted features, for example, a human visual model adapted for a 4×4 DCT block [16], relations between predicted DCT coefficients and real DCT coefficients [17]. Watermarking method proposed by Qiu et al. [18] embeds a robust watermark into the DCT domain and a fragile watermark into motion vectors during H.264 compression. J. Zhang and A.T.S. Ho [19] proposed a scheme that uses the tree-structured motion compensation, motion estimation and Lagrangian optimization of the H.264 standard. The authentication information is represented by a binary watermark sequence and embedded into video frames. Dima Pröfrock et al. [20] proposed a new transcoder, which analyses the original H.264 bit stream, computes a watermark, embeds the watermark for hard authentication and generates a new H.264 bitstream. All of these techniques either embed watermarks during the encoding process of the H.264 video [18][19] or employ cascaded decompression and recompression operations [16][17][20] to analyze H.264 bitstream and embed the watermark.

From the above discussion, we can notice that the literature contains extensive research on video adaptation, video encryption and video authentication approaches. However, a complete system entailing adaptation, encryption and authentication in one single framework operating in the compressed domain is lacking. Our proposed 3-in-1 system

for H.264 video, conforming to MPEG-21 DIA, is a realistic inspiration where any intermediary MPEG-21 compliant host can adapt, encrypt and authenticate the video without requiring prior knowledge of the video codec or bitstream.

IV. The Video Processing System

A systematic procedure for designing video adaptation framework involves identifying adequate entities (e.g. pixel, frame etc.) for adaptation as well as identifying a feasible adaptation technique (e.g. re-quantization, frame dropping etc.) [21]. In our temporal adaptation framework, I-frames and P-frames in each frameset[1] are the entities; and skipping frames from the original compressed bitstream dynamically is the chosen adaptation technique. Our design decision behind choosing this technique is made with the goal of creating a simple and computationally efficient adaptation engine. On the other hand, encryption is a process of scrambling or converting data known as Plaintext into a form, called a Ciphertext that cannot be interpreted by an unintended user or receiver. For authentication, a customized digital signature is inserted in a suitable marking space in the content itself. Now, to encrypt and authenticate a video content in the compressed domain requires the plaintext and the marking space to be selected in the compressed domain dynamically. The system performing the above three operations is shown in Figure 3. In this section, we explain the adaptation part in detail, and leave encryption and authentication for section V.

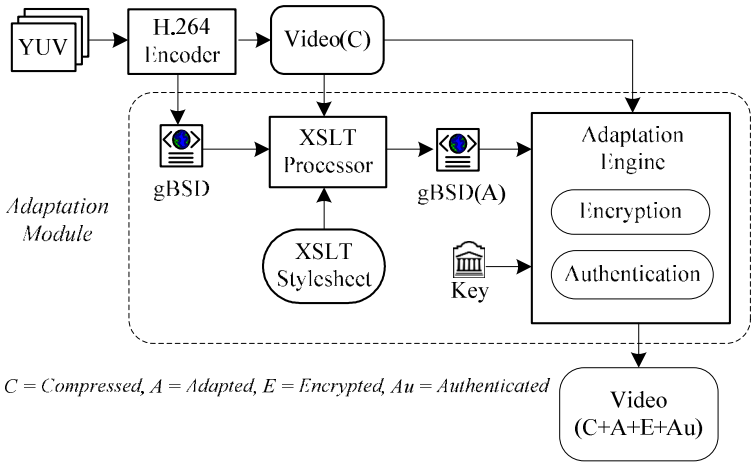

Figure. 3. Proposed architecture for adaptation, encryption and authentication

A. Generation of Digital Item

The DI (video bitstream along with its gBSD) is the basic content for resource server or content provider on the delivery path. In an MPEG-21 framework, the generation of original Bitstream Syntax Description from binary data (BintogBSD) is not normatively specified. In our approach, gBSD is generated during the encoding process of the bitstream because the encoder best knows the structure of the bitstream. For our prototype, we have enhanced the ITU-T reference software implementation JM 9.5 [22] with gBSD generation functionality. The gBSD consists of frame number, frame start, frame type and length of each frame for temporal adaptation. Considering that B frames are not usually used for motion compensation and reconstruction of other frames, configuration can be set to encode the raw YUV frames as only Intra frame (I-frame) and Inter frame (P-frame) in the encoder control. It is obvious that more I-frames in each frameset would increase the video quality at the cost of increased file size. In our system, in every frameset before and after adaptation, the very first frame is always an I-frame. This I-frame works as the

[1] For implementation convenience, 'Frameset' has been assumed to represent the number of frames equal to the frame rate at which the raw video is being encoded (usually 30fps).

reference frame and is used for random access. FrameSkip (number of frames to be skipped in input) and FrameRate (frame rate per second) for the original H.264 bitstream are left as variable. The final output from the H.264 encoder is the gBSD and encoded H.264 video, which together shape the Digital Item for adaptation. If B-frames are used then the frame skipping procedure (described in Adaptation Module later) also considers these frames to compute the frames that need to be dropped for achieving the target frame rate, provided B-frames are included in the computation of Frameset.

B. Adaptation Module

In the adaptation module, the adaptation process is completed in 2 steps – gBSD transformation characterizing the resulting media bitstream via XSLT [23], and video bitstream transformation based on the transformed gBSD. For the gBSD transformation, adaptation characteristics for the adapted gBSD are formed in an XSL style sheet[2] which is fed to the XSLT processor to transform the original gBSD. The XSLT processor takes a tree structure as its input by parsing the gBSD and generates another tree structure as its output into adapted gBSD. Template rules assembled in the style sheet are predisposed to filter the original gBSD based on the actual encoding frame rate and target frame rate. A frame skip pattern based on the original frame rate and required frame rate is devised to identify the appropriate nodes in the source tree of the gBSD. If the original encoding frame rate is 30frames per second (fps), mechanism enables a consistent frame-dropping pattern for a target frame rate in between 1 to 29fps. For example, for any DI, if the target frame rate is 10fps, then the selected frame numbers to be dropped will be same for all DIs. The pseudo-code of this strategy is shown below:

```
FrameSkip(newFrameRate,oldFrameRate)
{
 a=1, b=1;
 temp = newFrameRate/oldFrameRate;
 if(newFrameRate<oldFrameRate && newFrameRate!=0)
 {
  while(a<oldFrameRate)
  {
    copyFrame(a);
    a = ceiling(1+b*temp);
    b++;
  }
 }
}
```

The next step is the generation of the adapted bitstream using the transformed gBSD. Adapted bitstream generation consists of 3 steps - parsing the adapted gBSD, extracting the parsed gBSD information from the video and writing the adapted video stream to a buffer file. In every frameset, we ensure 1 I-frame for every 9 consecutive frames, which leads to having 3 I-frames for each second of video (and not only one I-frame). To handle multiple frame dependency, if the reference frames (I-frames or P-frames) are dropped then the very last available I-frame of that frameset is used as the reference frame. In the case of low target frame rate (i.e. less than 10fps), the very first I-frame in that frameset is considered as the reference frame. This avoids any perceptible distortions due to frame dropping. The final output is the adapted H.264 format compliant bitstream.

C. Live Adaptation

For XSLT, complete XML description must be loaded before being adapted. This is a shortcoming of applying the above adaptation architecture in live scenarios. To overcome this shortcoming and to offer live adaptation on top of the aforementioned implementation, we propose to process the live video streams as small clips in a pseudo live fashion. All the clips are encoded in H.264 format and have their own gBSD. These gBSDs are adapted for temporal adaptation and thus applied to gBSDtoBin process to serve a specific adaptation request. To process live streams,

[2] Style sheet defines the template rules and describes how to display a resulting document.

significant considerations need to be given to the frame rate and resolution of the captured frames because the processing time greatly varies for the higher frame rate and resolution. Moreover, depending on the webcam, video clips might need type conversion and pre-processing for specific resolution before putting it in the adaptation module to create DIs, adapt and transmit. As a result, streaming is vulnerable to an obvious fixed delay at the beginning, for pre-processing and adaptation of clips. For presentational applications, longer initial delays are usually acceptable in this type of live adaptation.

D. Evaluation of the Adaptation Module

The implementation scheme applying MPEG-21 DIA presented above is purely a temporal one, so the quality degradation of the adapted video is due to frame dropping only. The very first I-frame in each frameset is always set aside to be used as the reference frame for low target frame rates. To evaluate the prototype, an Intel P4 3.4Ghz Win XP Pro SP2 with 1GB RAM PC was selected as the media resource server. Table I shows the DI generation performance for the pre-coded video adaptation of four test sequences – Airshow, Carphone, Container and Corvette. Each of the video sequences consists of 300 frames of frame size CIF (352×288), QCIF (176×144) and SQCIF (128×96). In the encoder control, for different frame sizes, same configuration setting of different parameters (e.g. period of I-frames, number of reference frames, search range, bit depth for luminance and chrominance etc.) is used. Here we can see that, for live video adaptation, SQCIF frame size is appropriate because the DI generation performance for SQCIF video is greater than 15 fps.

TABLE I
DIGITAL ITEM GENERATION PERFORMANCE

Resolution	Airshow	Carphone	Container	Corvette
SQCIF	11.976s (@25.68fps)	11.628s (@25.23fps)	11.228s (@25.73fps)	11.761s (@25.51fps)
QCIF	23.640s (@12.69fps)	23.169s (@12.95fps)	23.251s (@12.80fps)	24.162s (@12.42fps)
CIF	99.190s (@3.02fps)	94.893s (@3.16fps)	97.152s (@3.09fps)	98.288s (@3.05fps)
Frame Rate : 30 fps, Intra Period: 9, Total Frames: 300, QP = 28				

TABLE II
TEMPORAL ADAPTATION PERFORMANCE: COMPRESSED DOMAIN (PROPOSED) VERSUS CASCADED

Resolution	New FPS	Airshow		Carphone		Container		Corvette	
		Proposed	Cascaded	Proposed	Cascaded	Proposed	Cascaded	Proposed	Cascaded
SQCIF	5	0.597s	32.124s	0.435s	32.062s	0.419s	32.248s	0.653s	32.389s
	10	0.825s	34.010s	0.798s	33.851s	0.659s	34.019s	0.928s	34.567s
	15	1.080s	36.053s	0.968s	35.534s	0.832s	36.032s	1.219s	36.361s
QCIF	5	0.569s	36.648s	0.594s	36.782s	0.578s	36.698s	0.972s	37.031s
	10	0.968s	40.268s	1.106s	40.856s	1.172s	40.538s	1.358s	41.007s
	15	1.438s	59.001s	1.469s	44.484s	1.765s	44.362s	2.016s	45.236s
CIF	5	1.032s	58.079s	0.950s	62.879s	0.719s	63.323s	1.878s	60.402s
	10	2.031s	77.884s	1.516s	78.553s	1.422s	79.819s	3.110s	78.751s
	15	3.094s	96.112s	2.280s	94.322s	2.157s	98.749s	4.688s	97.424s

Table II presents the temporal adaptation performance of the proposed adaptation architecture compared to that of a cascaded approach for the above four video sequences. In the cascaded approach, the decoder available in the ITU-T sample implementation JM 9.5 is used to decode the video; then the adapted video is re-encoded after the necessary adaptation (i.e. frame dropping to achieve target frame rate). From Table II, we can see that temporal adaptation time differs for different video sequences in terms of resolution and required frame rate. As can be seen, our approach significantly outperforms the cascaded approach. Table III & Table IV gives an average time profiling of the adaptation operations for both the proposed and the cascaded approaches respectively. Even though adaptation

in the compressed domain has a larger footprint than the cascaded approach in terms of the encoding (i.e. DI generation) and the adaptation procedures, it is considerably faster during the actual adaptation process (as can be seen in Table II). From a cost-benefit standpoint, since encoding will be done only once in a certain video's life time and adaptation will be done many times (for different user contexts) as needed, the compressed-domain approach is much more efficient. It is obvious that higher target frame rate necessitates relatively higher computation time because the adaptation engine processes more frames and their relative dependencies.

TABLE III
ADAPTATION TIME PROFILING OF THE VIDEO SEQUENCES (PROPOSED APPROACH)

Resolution	New FPS	*gBSD Transformation*			*Video Adaptation*	
		Original gBSD Parsing	Stylesheet Parsing	XSLT Transformation	Adapted gBSD Parsing	Adapting Video
SQCIF	5	*5.69 %*	*2.94 %*	*4.52 %*	*1.98 %*	*86.50 %*
	10	*3.66 %*	*1.88 %*	*4.12 %*	*1.81 %*	*83.67 %*
	15	*2.88 %*	*1.48 %*	*3.78 %*	*1.71 %*	*83.42 %*
QCIF	5	*4.62 %*	*2.38 %*	*6.49 %*	*1.34 %*	*86.01 %*
	10	*2.77 %*	*1.40 %*	*7.41 %*	*1.05 %*	*87.95 %*
	15	*1.94 %*	*0.97 %*	*7.80 %*	*1.18 %*	*88.84 %*
CIF	5	*2.64 %*	*1.35 %*	*7.79 %*	*0.84 %*	*90.84 %*
	10	*1.49 %*	*0.79 %*	*10.49 %*	*0.66 %*	*93.00 %*
	15	*1.04 %*	*0.51 %*	*12.01 %*	*0.64 %*	*94.15 %*

TABLE IV
ADAPTATION TIME PROFILING OF THE VIDEO SEQUENCES (CASCADED APPROACH)

Resolution	New FPS	*Decoding*	*Adapting Video*	*Re-encoding*
SQCIF	5	*93.72 %*	*0.05 %*	*6.23 %*
	10	*88.48 %*	*0.07 %*	*11.44 %*
	15	*83.86 %*	*0.09 %*	*16.05 %*
QCIF	5	*89.24 %*	*0.07 %*	*10.69 %*
	10	*80.74 %*	*0.08 %*	*19.18 %*
	15	*73.52 %*	*0.09 %*	*26.39 %*
CIF	5	*77.83 %*	*0.38 %*	*21.79 %*
	10	*60.33 %*	*0.41 %*	*39.27 %*
	15	*49.18 %*	*0.58 %*	*50.24 %*

TABLE V
LIVE ADAPTATION PERFORMANCE

Captured Video	Type	AVI To YUV	YUV To DI	Adapt	Adapted Video
SQCIF (128x96)	AVI	1.5 s	7.5 s	1.5 s	SQCIF (128x96)
15 fps					1-14 fps
Length 20 s					Length 20 s

Table V shows the adaptation performance of the proposed adaptation system for a live video session. Notice that AVI is used here simply due to the output format of the webcam in use. If the camera outputs YUV directly, then the delay will be even less. From this table we can infer that there will be an initial delay for the live video processing before transmission which will be around 30 seconds for a 20-second buffer. The initial delay is computed as follows:

Total Initial Delay = Capture time + AVItoYUV conversion time + YUVtoDI generation time + Adaptation time

This initial delay will not adversely affect presentational applications, such as news services or sports broadcasting,

since some reasonable initial delay is acceptable in such systems. If a smaller initial delay is needed, say 10 or 5 or even 1 second, it can be accommodated. However, for conversational applications involving synchronous collaboration, such as video conferencing, this delay could potentially cause a problem and needs more research.

V. Encryption and Authentication of adapted video

Encryption is used in many video transmission applications to ensure that only authorized receivers can access the media. Reasons for encryption include security, privacy, business reasons (only the subscribers who have paid should be able to view), age restrictions, and others. Therefore, this paper further presents that, with the help of the adapted gBSD, either macroblocks containing the motion vectors or slice data partitions of selective frames can be encrypted after adaptation. Content authentication on the other hand is an ongoing and constant requirement for the transmission of sensitive video content where integrity is a big issue. To embed copyright information, a robust watermark is required whereas for authentication, fragile or semi-fragile watermarks are sufficient. Generally, fragile watermarking systems or hard authentication rejects any modification made to a digital content. The idea is to authenticate the digital data by a hash-value. The hash value can be protected with a key and the key can be verified from a trust centre. Moreover, in most video adaptation delivery and consumption chains, watermark embedding and detection need to be performed in real time. For still picture, detection of the robust watermark can take as long as a few seconds but this delay is unacceptable for motion pictures especially when the frame rate is faster than given thresholds say more than 10fps. Hence, we extend the functionality of our framework by adding an authentication system which utilizes the adapted gBSD to select the marking space and to embed the authentication bits directly in the compressed video bitstream after adaptation.

A. Encryption of the Adapted Video

Here, we perform two types of encryptions on the video: 1) encrypting selective macroblocks, and 2) encrypting selective slice data partitions. These are explained next.

1) Encrypting selective macroblocks: In this method, macroblocks containing the motion vectors in each frame are taken as logical units. The encryption engine considers the level of encryption based on the user's preference. If the encryption preference is high then all the frames in the adapted video are encrypted; else, either only the I-frames or all frames from the selected framesets are encrypted. For encryption, first, the frame marker is scanned from the gBSD. Thus, each macroblock's starting position and corresponding length is retrieved from the transformed gBSD for those frames which need to be encoded. To encrypt the macroblocks, an encryption key is chosen and an XOR operation is performed. After the XOR operation, the bitstream is processed as usual for H.264 format compliance. It is worth mentioning that each logical unit can be encrypted independently.

2) Encrypting selective slice data partition: From Section II we learnt that in H.264 slice sizes are flexible. In this method, we consider a frame to be partitioned into multiple slices which will be the logical units for encryption. The encryption engine considers the level of encryption based on user preferences like that of encrypting the macroblocks. Initially for the encryption, the frame marker, start and length positions are parsed from the adapted gBSD for all frames. Thus, each of the slice partitions is retrieved from the transformed gBSD for those frames, which need to be encoded. To encrypt/scramble the selected slice data for perceptual degradation, again an encryption key of arbitrary length is chosen and an XOR operation is performed on 'luminance' and 'chrominance' component coefficients. It is important to mention that each of the logical units can be encrypted independently with a different key. The final output after macroblock/slice encryption is the adapted and encrypted video which is H.264 format compliant. Therefore, a standard H.264 player would be able to decode and play the video, but encrypted frames will not be perceived clearly unless decrypted with the right key.

3) Experimental Results: The resulting encrypted video from the above two methods have frames which are encrypted and would not be visually perceived clearly. Four frames of the results from the sample videos are shown in Figure 4 for selective macroblock encryption. From Figure 4.h., we can observe that macroblock encryption is not fairly useful for a video (or video frames) which have mostly static objects in the scene. For such case, we

recommend to use the slice data encryption as shown in Figure 5. For selective slice data encryption, the encrypted slice contains visual artifacts hiding the actual visual data, which is an interesting technique to hide visual information without removing or hampering other slice data partitions of the respective frame. As can be seen, for both the methods, the resulting frames are visually distorted and not visibly useful without decryption.

a. Carphone - *Original* b. Carphone - *Encrypted* c. Airshow - *Original* d. Airshow - *Encrypted*

e. Corvette - *Original* f. Corvette - *Encrypted* g. Container - *Original* h. Container - *Encrypted*

FIGURE. 4. SAMPLE MACROBLOCK ENCRYPTED VIDEO FRAMES

Container - *Original* Container - *Encrypted*

FIGURE. 5. SAMPLE SLICE DATA ENCRYPTED VIDEO FRAME

B. Authentication of the Adapted Video

To present a watermarking based authentication scheme in the compressed domain, we have investigated the possibility to utilize gBSD to select the marking space in the compressed domain where a digital signature (watermark) can be embedded. The investigation result gives us a direction to perform hard authentication of the adapted video to assure integrity. We make use of the gBSD to identify the segments into which the authentication bits can be embedded.

1) Selecting the Marking Space: From the gBSD, a marking space can be selected from available alternatives, like frame, slice, macroblock, and block. An application specific marking space can be selected in a predefined way and a fixed watermark embedder can be designed. Otherwise, if the marking space is selected manually, the watermark embedder should be capable of inserting watermark bits in the selected segment directly in the compressed bitstream.

For manual selection of the marking space, start and length of each segment need to be defined in the gBSD. At the same time, selection of a marking space and applying customized modification must conform to the H.264 bitstream specification to assure compliance for a standard player or decoder. In our implemented system, we have made use of the slice data to compute authentication bits and finally embedded these bits in the slice header.

2) Watermark embedder: The watermark embedder module is designed to embed authentication bits on the fly while adapting as shown in Figure 6. An arbitrary private key will be an input to the adaptation engine to perform the necessary computation of the authentication bits.

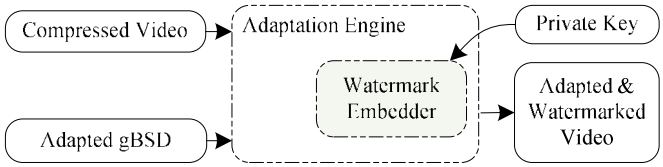

FIGURE. 6. ADAPTATION AND WATERMARKING MODULE

In the adaptation engine, while adapting the video bitstream, we embed the authentication bits in the slice header VLC byte align bits (minimum 1 bit and maximum 7 bits). It is important to mention that this marking space can be further extended to other entities like frame and macroblock based on the gBSD details. Total number of bits in a slice (S_N) is the sum of slice header ($S_{H(N)}$) bits and slice payload ($S_{P(N)}$) bits, denoted as, $S_N = S_{H(N)}+S_{P(N)}$, where N = *total number of bits*. From the gBSD, length of the slice header VLC byte align bits (*vlcn*), start and length of the frame are parsed. A hash value (F_{Hash}) of the frame data including slide header (except the bits where the authentication bits will be embedded) and slice payload is computed. The architecture applies a simple hash function based on PJW Hash [24] which can be replaced by any available advanced hash function. Input to the hash function is the frame data, length of frame data (in bytes) and a private key (P_K). For implementation purpose, we have considered a logo/sample image of an arbitrary length (L_N) as our private key. Authentication bits embedded in the slice header (S_H) can be denoted as follows:

$$S_{H(N-vlcn+j)} = F_{Hash}(j) \oplus P_{K(i)} \quad where, 1 \le j \le vlcn, 1 \le i \le L_N$$

After embedding the authentication bits, an optional second level of authentication is applied by scrambling the last byte of the slice payload to restrict re-computation of the signature by an intruder. In H.264, blocks and macroblocks are not byte-aligned, so an XOR operation is applied to the last byte of slice payload (S_{lbsp}) with respect to the private key like that of slice header. Even though to re-compute the authentication bits, along with the hash value, the private key is necessary, the modified slice payload will add another layer of assessment to detect possible attacks. Modification made to the slice payload can be shown as:

$$S_{lbsp} = S_{lbsp} \oplus P_{K(i)} \quad where, 1 \le i \le L_N$$

3) Watermark Detector: The watermark detector consists of a 4-step process. The first step, parsing adapted gBSD, is extracting the marking space from the adapted gBSD to identify each watermarked segment. The second step, restore frame data, comprises of XOR-ing the scrambled slice payload bits with the private key to re-instate the slice data for computing original hash value. The third step, watermark extraction, extracts the authentication bits from slice header. The final step corresponds to compare the computed hash value from slice data with the extracted value from slice header.

To verify a received video content, the user needs, in addition to the video data, the private key and the adapted gBSD. There are many approaches for the secure transmission of private key and gBSD and this topic is beyond the scope of this research. In case of a video content for mass distribution without any priority given to authentication, it is not necessary to modify the decoders for every client so that the adapted gBSD need not be transmitted to the receiver. In the latter case, typical H.264 players will be able to play the content without any prior knowledge of the modifications made to the content since the content structure conforms to the H.264 bitstream syntax structure.

4) Experimental Results: The benefit of embedding the authentication module inside the adaptation system is that

it reduces the overhead to parse the adapted gBSD for authentication. Figure 7, shows a comparative analysis of the 3-in-1 system. Here we can see the performance of the encryption module and the watermarking module on top of the adaptation system for pre-recorded videos. The time required to embed authentication bits is a little higher than the adaptation and the encryption time. The difference between these two depends on the hash function used. More complicated or robust hash functions will require higher execution time to compute and embed the authentication bits. Another factor that can affect the execution time is the marking space. To make the system more robust, one can decide to embed the authentication bits in the frame, macroblock and block, which will eventually require longer watermarking time.

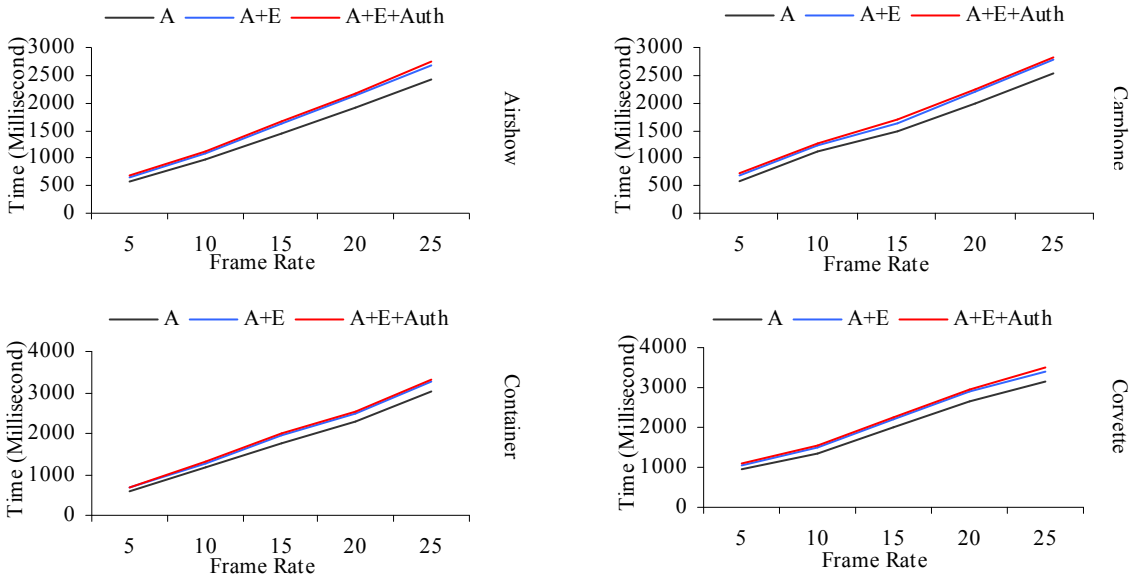

FIGURE. 7. OVERALL SYSTEM PERFORMANCE

To perform encryption (and/or adaptation) operations in intermediary nodes, it is important to transmit the corresponding (adapted) gBSD together with the (adapted) video bitstream. In this case, once a specific video request is made, both the files (video and gBSD) need to be transmitted from the media resource server. To decrypt the encrypted bitstream, the decryption module of the end node needs to have the adapted gBSD to identify the encrypted slice partitions along with the decryption key.

VI. DISCUSSION

We have seen so far a framework for the utilization of MPEG-21 gBSD to adapt, encrypt and authenticate video data in the compressed domain. A multi point adaptation on the encoded bitstream is achievable as long as the bitstream description is preserved for the new adapted stream conforming to the gBS schema. The achieved benefit of the work is that the end devices are free of any adaptation operations. Moreover, to protect the security and integrity of the adapted media during transmission, we showed that knowledge of the bitstream along with content structure in the form of metadata enables the encryption and the authentication processes to be customized and easily deployable in the adaptation engine. However, the decryption and the authentication modules are optional for the end-node devices and needs not to be implemented in all end-nodes unless it is obligatory. Depending on the application scenario, if the decryption and/or verification of the video are insignificant (e.g. no encryption/authentication have been made), then the transmission of the corresponding gBSD data can be omitted.

For decrypting and authenticating the received video, the end node should have a decryption and authentication module embedded in the player respectively. Certainly, metadata delivery and processing for decryption and authentication of video data leads to an additional overhead but from our experimental results we have found that the size of the gBSD is around 2% of the encoded video file size for temporal adaptation.

Since the capabilities of small handheld devices are limited, to process received compressed domain video data in these devices, video data can be transmitted in small chunks along with the corresponding gBSD. The gBSD parser identifies the sections from the gBSD where the encryption and authentication operations are performed first. The respective modules thus operate only on those limited sections of the corresponding video data (e.g. a frame or a frameset depending on the processing power and memory of the device). It is noteworthy that, the purpose of the aforementioned encryption framework is not to provide a full-proof encryption technique or algorithm. The proposed framework applies the MPEG-21 DIA, where encryptable plaintext and/or marking space is identified from the gBSD (which is parsed only once while adapting) as opposed to applying the encryption and authentication on the whole bitstream, which eventually reduces the cost of computation and processing. Since decompression is less complex than compression, the overhead added to decrypt the encoded macroblocks or slices would not exceed the nominal threshold value for presentation, as they will reside as a closely coupled task. For authentication, the original H.264 video is not required; rather a separate authenticator can verify the validity of the received video data. The authenticator is independent of the decoder, so there will be no lag added while decoding the video. Instead of computing hash value for every frame, a digital signature can be computed for the whole video content or for a certain number of frames (e.g. frame rate at which the raw video is being adapted) and thus embedded in the selected marking space.

In an UMA environment, a Media Streaming Server (MSS) can be connected to several Proxy Servers (PS) and clients. The MSS can serve clients which are directly connected to it. It sends the adapted video content, adapted gBSD and the private key to the client(s) accordingly. In response to a request from a PS (e.g. PS 02) for a video content, MSS will send the original H.264 video and the gBSD. PS 02 will then serve the subordinate clients with proper adaptation and watermarking. Another PS (e.g. PS 01) can serve only classified clients (e.g. small handheld devices) and thus request for a specific adapted video (e.g. SQCIF, 15fps) from the MSS. In this scenario, the MSS will send the specified adapted video content along with the adapted gBSD (with an optional watermarking). PS 01 will then adapt and re-compute the authentication signature before serving a client. As we can see, none of the servers needs to be aware of the codec. It can adapt and watermark the videos on the fly just looking at the gBSD.

VII. Conclusion

Seamless adaptation and transcoding techniques to adapt digital content have achieved significant attention due to the need to serve consumers with the desired content in a feasible way. Two desirable characteristics of an adaptation system for video streaming can be noted as adapting the media and protecting the security of the media. Most recent commercial video monitoring/surveillance tools and applications intend to apply H.264 format for video due to its flexibility and quality. Video data being captured by wireless/dispersed cameras and transmitted to a distant receiver thus requires embedding a signature in real time to scrutinize the integrity of the received content in a contained environment. In this work, we showed that knowledge of the bitstream along with content structure in the form of metadata enables the encryption and authentication processes to be customized and integrated in the adaptation system. Beyond the developments presented above, interleaved video and audio bitstream adaptation in real time is an emergent field of research. For the time being, our notion for audio is to transmit the audio bitstream in a separate channel where starting of each frameset in video bitstream would be the synchronization point while playing. Scalable Video Coding (SVC), part 13 of the Multimedia Framework for a pervasive media application environment is another interesting area of study. Moreover, in pervasive networking environment a reasonable solution is required that will disallow possibly untrusted intermediary adaptation engines in the delivery path to adapt content in the encrypted domain. Further research to embed robust watermark in compressed domain for copyright protection purposes is an interesting research field for the current research team. Finally, to avoid the obvious fixed delay while

capturing and processing the DIs, a hardware level implementation capable of generating the compressed bitstream and gBSD will definitely faster the process.

REFERENCES

[1] R. Iqbal, S. Shirmohammadi, and C. Joslin, "MPEG-21 Based Temporal Adaptation of Live H.264 Video," *in Proc. of IEEE Intl. Symposium on Multimedia*, Dec. 11-13, 2006, pp. 457-464.

[2] R. Iqbal, S. Shirmohammadi, and A. El Saddik, "A Framework for MPEG-21 DIA Based Adaptation and Perceptual Encryption of H.264 Video," *in Proc. of SPIE/ACM Multimedia Computing and Networking Conference*, Vol. 6504, Jan. 28 - Feb. 1, 2007, pp. 650403-1 – 650403-12.

[3] ISO/IEC 21000-7:2004, Information Technology – Multimedia Framework – Part 7: Digital Item Adaptation.

[4] A. Vetro and C. Timmerer, "Digital item adaptation: overview of standardization and research activities," *IEEE Trans. on Multimedia*, Vol. 7, Issue 3, Jun. 2005, pp. 418-426.

[5] L. Rong and I. Burnett, "Dynamic multimedia adaptation and updating of media streams with MPEG-21," *in Proc. of First IEEE Consumer Communications and Networking Conference*, 2004, pp. 436-441.

[6] S. Wenger, "H.264/AVC over IP," *IEEE Trans. on Circuits and Systems for Video Tech.*, Vol.13, Issue 7, 2003, pp. 645-656.

[7] C. Gomila and P. Yin, "New features and applications of the H.264 video coding standard*," in Proc. of Intl. Conf. on Info. Tech.: Research and Education*, 2003, pp. 6 – 10.

[8] J. Xin, C.-W. Lin, and M.-T. Sun, "Digital video transcoding," *Proc. IEEE*, vol. 93, no. 1, Jan. 2005, pp. 84–97.

[9] K. Taehyun and M.H. Ammar, "Optimal quality adaptation for scalable encoded video," *IEEE Journal on Selected Areas in Communications*, Vol. 23, 2005, pp. 344-356.

[10] M.A. Bonuccellit, F. Lonetti, and F. Martelli, "Temporal transcoding for mobile video communication," *in Proc. of Second Annual Intl. Conf. on Mobile and Ubiquitous Systems: Networking and Services*, 2005, pp.502-506.

[11] F. D. Vito, T. Ozcelebi, R. Civanlar, A. M. Tekalp, and J. C. D. Martin, "Rate Control For GOP-Level Rate Adaptation in H.264 Video Coding," *in Proc. of Intl. Workshop on Very Low Bit-rate Video Coding*, 2005.

[12] I. Shin, Y. Lee, and H. Park, "Motion estimation for frame-rate reduction in H.264 transcoding," *in Proc. of Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems*, 2004, pp.63-67.

[13] S. Devillers, C. Timmerer, J. Heuer, and H. Hellwagner, "Bitstream syntax description-based adaptation in streaming and constrained environments," *IEEE Trans. on Multimedia*, Vol. 7, Issue 3, Jun. 2005, pp. 463-470.

[14] H. Cheng and Li Xiaobo, "Partial encryption of compressed images and videos," *IEEE Trans. on Signal Processing*, Vol.48, Issue 8, 2000, pp. 2439-2451.

[15] T. Lookabaugh, "Selective encryption, information theory and compression," *In Proc. of Conf. Record of the 38th Asilomar Conf. on Signals, Systems and Computers*, Vol.1, 2004, pp. 373-376.

[16] M. Noorkami and R. M. Mersereau, "Towards Robust Compressed-Domain Video Watermarking for H.264," *SPIE Security, Steganography, and Watermarking of Multimedia Contents*, Vol. 6072, Jan. 2006, pp. 489-497.

[17] G. Wu, Y. Wang, and W. Hsu "Robust Watermark Embedding Detection Algorithm for H.264 Video," *Journal of Electronic Imaging*, Vol. 14, Jan.-Mar. 2005.

[18] G. Qiu, P. Marziliano, A.T.S. Ho, D. He, and Q. Sun, "A Hybrid Watermarking Scheme for H.264/AVC Video", *in Proc. Of 17th Intl. Conf. on Pattern Recognition,* Vol. 4, 2004, pp. 865-869.

[19] J. Zhang and A.T.S. Ho, "Efficient Video Authentication for H.264/AVC," *in Proc. of 1st Intl. Conf. on Innovative Computing, Information and Control*, Vol. 3, Aug. 2006, pp. 46-49.

[20] D. Pröfrock, H. Richter, M. Schlauweg, and E. Müller, "H.264/AVC Video Authentication Using Skipped Macroblocks for an Erasable Watermark," *in Proc. of SPIE Visual Communications and Image Processing*, Vol. 5960, 2005, pp. 1480-1489.

[21] S-F Chang and A. Vetro, "Video Adaptation: Concepts, Technologies, and Open Issues," *Proc. IEEE*, Vol. 93, No. 1, Jan. 2005, pp. 148 – 158.

[22] http://ftp3.itu.ch/av-arch/jvt-site/reference_software/

[23] http://www.w3.org/TR/xslt

[24] A. V. Aho, R. Sethi, and J. D. Ullman, Compilers: Principles, Techniques, and Tools, Addison-Wesley, pp. 434-438, 1986.