NOTE TO USERS

uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Ziad Sakr

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Designed System for the Composition and Protection of MPEG-4 and SMIL Scenes and their Multimedia Content

TITRE DE LA THÈSE / TITLE OF THESIS

Nicolas Georganas

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Abdulmotaleb El Saddik

Dimitrios Hatzinakos

Evangelos Kranakis

Jiying Zhao

Gary W. Slater

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES /
DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

Designed System for the Composition and Protection of MPEG-4 and SMIL Scenes and their Multimedia Content

by

Ziad Sakr, M.A.Sc. in Electrical Engineering

A thesis submitted to the
Faculty of Graduate and Post-Doctoral Studies
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical & Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

School of Information Technology and Engineering
University of Ottawa

Your file Votre référence
ISBN: 0-494-11018-X
Our file Notre référence
ISBN: 0-494-11018-X

Abstract

For the past decade, there have been numerous research works focusing on the protection of digital images, audio, video, 3D virtual scenes, and software data from unauthorized use and distribution. With the emerging technology of the MPEG-4 standard, many different media objects, such as images, video, audio, and 3D virtual objects, may be combined together to compose a given MPEG-4 scene. A MPEG-4 scene can be composed using the MPEG-4 XMT textual-based format or the MPEG-4 BIFS compressed binary format. Using the MPEG-4 XMT standard, MPEG-4 scenes can easily be constructed. XMT allows content authors to exchange their content with other authors, tools, or service providers and facilitates interoperability with MPEG-4, X3D and SMIL. In order for owners and designers to protect and/or authenticate their work, some form of security needs to be applied to the MPEG-4 scenes and its media content.

This thesis presents the design of an MPEG-4 player that is able to compose or open a scene whether it is in the MPEG-4 BIFS format or the MPEG-4 XMT format. The scene can also be edited and modified with nodes or parts of the scene added, deleted, or translated in position with respect to the rest of the scene. All updates and modifications to the scene are viewed on the display scene graph. The MPEG-4 scene can also be saved in any of the desired formats whether the MPEG-4 BIFS for enhanced compression or the MPEG-4 XMT for ease of editing and modification.

The thesis also presents the design and implementation of a novel robust algorithm for the authentication and protection of MPEG-4 scenes and their media content. The algorithm is applied on the MPEG-4 XMT-A scenes. Even though the XMT structure lacks the noise components, which are considered a vital part for information hiding in media objects such as image, audio and video scenes, the designed and developed algorithm still proved to be robust against many forms of scene structure modifications

and tampering attacks. Even if a given scene is in the MPEG-4 BIFS format, it can be converted to the MPEG-4 XMT format and then checked for authenticity and any located copyrighted material.

The design and implementation of a new algorithm for SMIL scene authentication and media content location and copyright protection is also presented in this thesis. The test results shown proved the algorithm to be robust against different scene structural modifications and tampering attacks.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Nicolas D. Georganas for the encouragement, guidance and support he provided me throughout my PhD studies. He was always available to provide me with his knowledge, advice and help whenever I needed it. I couldn't ask for any better supervisor. Thank you very much.

I would also like to thank Dr. Abdulmotaleb El Saddik for providing me with great ideas at the beginning of my research that I was able to use throughout my work.

I also wish to thank Dr. Jiying Zhao for providing me with his time, knowledge and advice whenever I needed his opinion on my research work.

In addition, I would like to also acknowledge Francois Malric for his continuous administrative support.

Many thanks to all my good friends who provided with all the help and support whenever I needed it.

Last, but not least, I would like to give special thanks to my dad, mom, sisters, and brother, who each provided me with unique love, support, encouragement and patience throughout my studies.

Thank you very much.
Ziad Sakr
Ottawa, March 2005

Content

List of Acronyms

2D	Two Dimensional
3D	Three Dimensional
AAP	Association of American Publisher
A/D	Analog to Digital
ATM	Asynchronous Transfer Mode
BIFS	BInary Format for Scenes
BMP	Bitmap format
CIDf	Content ID Forum
Codec	Encoder/Decoder
CPTWG	Copy Protection Technical Working Group
D/A	Digital to Analog
DCT	Discrete Cosine Transform
DES	Data Encryption Standard
DFT	Discrete Fourier Transform
DHSG	Data Hiding Sub Group
DMIF	Delivery Multimedia Integration Framework
DOI	Digital Object Identifier
DOM	Document Object Model
DTD	Document Type Definition
DWT	Discrete Wavelet Transform
GF	Galois Field

GIF	Graphic Interchange Format
GUI	Graphical User Interface
HAS	Human Audible System
HTML	Hyper Text Markup Language
HVS	Human Visual System
IA-DCT	Image Adaptive – Discrete Cosine Transform
IA-W	Image Adaptive - Wavelet
IDEA	International Data Encryption Algorithm
IEC	International Electrotechnical Commission
IP	Internet Protocol
IPMP	Intellectual Property Management and Protection
ISO	International Organization for Standardization
JAXP	Java API for XML Processing
JND	Just Noticeable Difference
JPEG	Joint Photographic Experts Group
LFSR	Linear Feedback Shift Register
LPM	Log Polar Mapping
LSB	Least Significant Bit
MB	Matching Block
MD5	Message Digest, Version 5
MEP	Macro Embedding Primitive
MPEG	Moving Picture Experts Group
MSB	Most Significant Bit
MSE	Mean Square Error
NISO	National Information Standards Organization
OeBF	Open Book Forum
PN	Pseudo Noise
PRBS	Pseudo Random Binary Sequence
PRMVS	Pseudo Random Multi Valued Sequence

PRS	Pseudo Random Sequence
PSNR	Peak Signal Noise Ratio
QoS	Quality of Service
RB	Range Block
RBS	Random Byte Sequence
RTP	Real Time Transport Protocol
SDMI	Secure Digital Music Initiative
SMIL	Synchronized Multimedia Integration Language
SNHC	Synthetic and Natural Hybrid Coding
TSPS	Triangle Strip Peeling symbol Sequence
TSQ	Triangle Similarity Quadruple
TVR	Tetrahedral Volume Ratio
UDP	User Datagram Protocol
VLC	Variable Length Codes
VRML	Virtual Reality Modeling Language
XML	Extensible Markup Language
XMT	Extensible MPEG-4 Textual format

Chapter 1

Introduction

The rapid development and deployment of new information technologies has improved the ease of access to digital information. It has also led to fears that copyright could be eroded by the illegal copying and redistribution of digital media. While equipment capable of copying audio, video and text content has long been available for domestic use, the loss of quality that analog copying entails, and the labor involved in the physical process of copy production has acted to limit copyright abuse. With digital media however, perfect copies can be produced and distributed with little effort. Modern compression algorithms have made copying even easier by the reduced size of those media files. This is a particular concern to commercial publishers of digital media content (e.g. audio, video) whose existence depends on defending the copyright of their information assets. The International Intellectual Property Alliance (IIPA) estimates the annual lost revenues in the U.S. motion picture industry due to piracy at US$1.3 billion, and for the record and music industries at US$1.7 billion [77]. If content owners cannot be assured that they will be properly compensated for use of their works, it will be unlikely that they will make these available for access over public networks. Therefore, mechanisms to protect content are seen as a necessary step towards the creation of a global commercial information infrastructure.

Conventional cryptographic systems permit only valid key holders access to encrypted data, but once such data are decrypted, there is no way to track their reproduction or retransmission. Therefore, conventional cryptography provides little protection against data piracy, in which a publisher is confronted with unauthorized reproduction of information. A digital watermark is intended to complement cryptographic processes [9]. It is a visible, or preferably invisible, identification code that is permanently embedded in the data and remains present within the data after any decryption process.

The concept of digital watermarking is derived from steganography [55], an ancient art of hiding information. Steganography, derived from Greek, literally means 'covered writing'. Throughout history, people have hidden information by a multitude of methods and variations. The heads of most trusted slaves were shaved and tattooed with a message that disappeared after the hair had regrown. Invisible ink made out of organic substances such as milk or urine, was also used extensively. Even information was put into paper by making the layer of pulp thicker or thinner, which can be viewed by placing the paper against the light or over a black surface. This type of information embedding is still used till now in identifying the manufacturer's signature, or as a security measure to avoid forgery of important documents such as bank notes, passports, etc. This is only a small list of many more steganographic techniques used for information hiding.

Both steganography and watermarking describe techniques that are used to imperceptibly convey information by embedding it into the cover-data. However, steganography typically relates to covert point-to-point communication between two parties. Thus, steganography methods are usually not robust against modification of the data, or have only limited robustness and protect the embedded information against technical modifications that may occur during transmission and storage, like format conversion, compression, or digital-to-analog conversion. Watermarking on the other hand, has the additional notion of resilience against attempts to remove the hidden data. Thus, watermarking, rather than steganography principles, is used whenever the cover-data are

available to parties who know the existence of the hidden data and may have interest removing them. A popular application of watermarking is to give proof of ownership of digital data by embedding a copyright statement. It is obvious that for this application the embedded information should be robust against manipulations that may attempt to remove it.

1.1. Basic Watermarking Principles

All watermarking systems consist of a watermarking embedding system for inserting a watermark into an object, and a watermarking extracting system for extracting a watermark from an object. Figure 1-1 shows the general watermark embedding scheme. The input to the scheme is the cover data used to embed the watermark in, the watermark, and an optional public or secret key used to enforce the security and prevent unauthorized users from recovering the watermark. The watermark can be of any nature such as a number, text, or an image. The output of the watermarking scheme is the watermarked data.

The watermark extraction system is depicted in Figure 1-2. Inputs to the scheme are the watermark or the original data, the private or public key, and the watermarked/tampered data. The system's output is either the recovered watermark W from the data or a confidence measure indicating whether the watermark has been detected in the object or not.

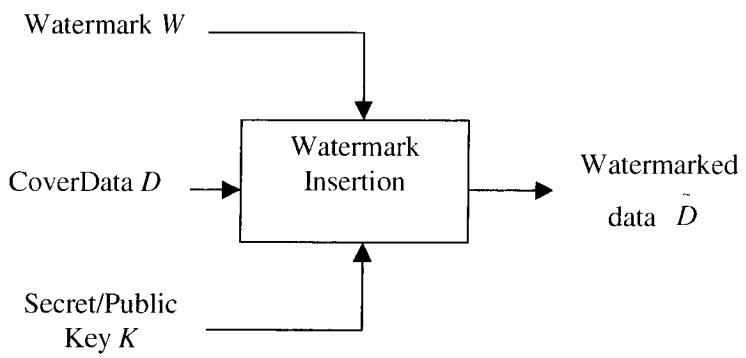


Figure 1-1 General watermark insertion scheme [55]

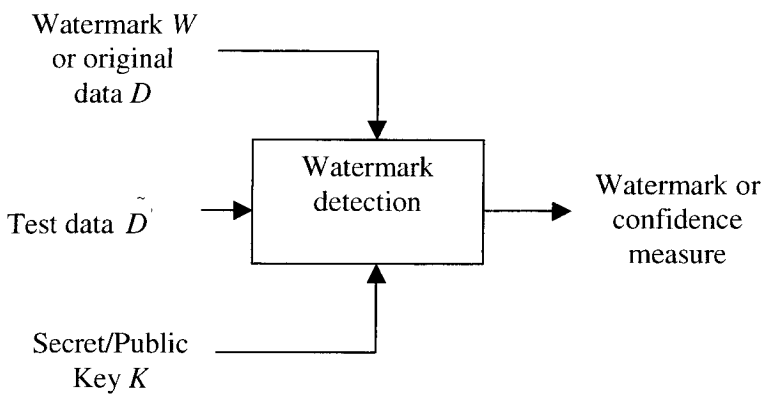


Figure 1-2 General watermark extraction scheme [55]

1.2. Properties of Digital Watermarks

Each watermarking application has its own specific requirements. Therefore, there is no set of properties that have to be met by all watermarking techniques. Nevertheless, some general directions can be given for most of the watermarking applications:

Perceptual transparency: The watermark should be imperceptible to the human observer, while maintaining the quality of the host data. Small modifications might

become apparent when the original data are compared directly with the watermarked data. But since users of watermarked data normally do not have access to the original data, they cannot do the comparison. Therefore, it is sufficient that the modifications in the watermarked data go unnoticed as long as the data are not compared with the original ones.

Robustness: A watermark is said to be robust, if it survives any kind of distortion introduced by standard or malicious processing such as: A/D and D/A conversion, lossy compression, affine transforms (translation, rotation, scaling and shearing), linear/nonlinear filtering, additive and multiplicative noise, multiple watermarking, statistical averaging, mosaic attack etc. The basic principle is to design a watermarking algorithm that is robust enough such that successful attacks would also impair the commercial value of the host data.

Watermark recovery with or without the original data: There has been a great research work where the original data are used to extract the watermark. But in some applications it may be impractical to use the original data to recover the watermark, such as in real-time video applications, or impossible to obtain in applications such as data monitoring or tracking. Some algorithms now are being developed to meet those requirements where the original data are not required for watermark extraction.

Computational cost: Different applications require the watermark to be done at different speeds and/or complexity. In broadcast monitoring, both embedders and detectors are required to work in real time. The embedders must not slow down the media production schedule, and the detector must keep up with the real time broadcasts. On the other hand, speed is not an issue when tracking illegal copies (robust watermark), watermark retrieval is needed only when copyright violations have to be investigated. Here the watermark insertion should be of low complexity, but the retrieval operation should be more complex in order to account for all possible kinds of attacks on the watermark.

Watermark security: Two types of watermark security can be identified: In the first type, an unauthorized user can neither read nor decode an embedded watermark nor can he detect whether a given set of data contains a watermark. The second set permits any user to detect if data are watermarked, but the embedded information cannot be read without having access to the secret key.

Resolving rightful ownership: In order to resolve rightful ownership of the watermark it must be possible to detect who was the first to watermark the data by imposing some constraints such as noninvertibility [22] of the watermark or using timestamps.

1.3. Classification of Digital Watermarks

Digital watermarks can be classified based on the application being inserted into, the nature of the watermark, the insertion technique used, etc.

Visible and Invisible watermarks: Visible watermarks are visible seals inserted into an object, i.e. image or video. They can easily be found when tuning to specific TV channel. They are located at the bottom right of the screen as a logo to identify the channel being watched. Those visible watermarks are mostly used to identify the copyright owner and not to withstand attacks. They are more prone to attacks than invisible watermarks due to their visibility.
Invisible watermarks are hidden from the viewer's eyes. They are more robust against attacks and can be used to identify the copyright owner when required.

Spatial and Spectral watermarks: Spatial watermarks are invisible watermarks inserted in the spatial domain. Most of the earlier watermarking techniques used to insert watermarks in the spatial domain. However, with the advance of compression techniques used on media objects, most of those spatial watermarks can easily be removed.

Spectral watermarks are usually more robust against compression algorithms than spatial watermarks. The watermark is embedded in the frequency domain instead of the spatial domain. Moreover, spectral watermarks can embed a large number of bits without incurring any noticeable visual effects, as opposed to spatial watermarks, which have low bit capacity to embed the watermark without being visible.

Robust and Fragile watermarks: Robust watermarks are mostly used to prove ownership of a media document and to detect any unauthorized copies. They should withstand any type of attack, such as compressing, cropping, rotating, printing, scanning, etc to a point that the document can still be declared useful.

Fragile watermarks on the other hand are designed to be fragile. Any minor tampering with the media document will break the watermark and declare the document useless. This type of watermark is made to prove content authentication and reveal that the document has not been tempered with when required.

Adaptive and Non-Adaptive watermarks: In non-adaptive watermarks, a constant scaling parameter α is used to insert the watermark. The watermarked object $V_i' = V_i + \alpha X$, where V_i is the media object before watermarking and X is the watermark. The scaling parameter determines the extent to which X alters V.

In adaptive watermarks, visual models are designed and used to insert the maximum amount of watermark into an object without degrading the quality of the object or making the watermark noticeable to the viewer. For instance, Just Noticeable Difference (JND) thresholds are based on the three different properties of the human visual system: frequency sensitivity, luminance sensitivity and contrast masking. They are used to determine the maximum amount of watermark signal that can be tolerated at every pixel location without affecting the visual quality of the image.

1.4. Applications of Watermarking

Different watermarking systems are developed based on the applications that are used for [76]:

Copyright protection: One of the main reasons for watermarking is for copyright protection. The idea is to embed information about the copyright owner into the data to prevent other parties from claiming to be the rightful owners of the data. The watermarks used for that purpose are supposed to be very robust against various attacks intended to remove the watermark. They also have to be unambiguous and still resolve rightful ownership, if other parties embed additional watermarks.

Fingerprinting: To avoid unauthorized duplication and distribution of publicly available multimedia content, an author can embed a distinct watermark (or fingerprint) into each copy of the document. The watermarks will be similar to serial numbers of software products. This is useful to monitor or trace back illegally produced copies of the data that may circulate around. However, the distribution of individually watermarked copies allow for collusion attacks. Therefore, such algorithms have to be designed in a way to be collusion-secure [11][12].

Copy protection: It is desirable in some systems to have a copy protection mechanism to disallow copying of the media content. A watermark can be used in such systems to prevent the copying of the media content or limit the number of copies to a certain number. For example, a compliant DVD player should not be allowed to playback or copy data that carry a "copy never" watermark. Data that carry a "copy once" watermark may be copied once, but no further copies should be allowed to be made.

Authentication: To be able to authenticate a document, any changes or tampering with the document should be detected. This can be achieve through the use of "fragile

watermarks" which have low robustness to certain modifications such as lossy compression but will be destroyed easily for any other modification applied to the document. Among all possible watermarking applications, authentication watermarks require the lowest level of robustness.

Content protection: In some applications where a free preview of a media content is provided to the general public, a visible watermark can be embedded into it to make the document commercially worthless. The watermark should be hard to remove without degrading the media quality.

1.5. Thesis Contributions

1.5.1. Accomplished Research Work

The Moving Picture Experts Group (MPEG) of ISO/IEC has put forward the MPEG-4 standard for the coding of audio-visual objects. In contrast to MPEG-2 and older standards, MPEG-4 is not a frame-based but an object-based framework. The objects can be audio and video data, 3D models, or other multimedia data. Moreover, the standard differentiates between two types of objects – natural objects (i.e. natural speech, images, music, or video created by camera or microphone) and synthetic objects (3D models and speech generators). The complete scene depends on the scene graph, which contains all the multimedia data and the detailed information for each object (timing, synchronization, movement, etc.) for the scene to be displayed properly.

All existing watermarking algorithms focus on the protection and authentication of specific media objects, such as image, audio, video. But with MPEG-4, one scene can contain multiple media objects of different types. To protect and authenticate such a scene, is a very challenging problem.

In this thesis work, a Codec has been developed, that is able to encode/decode a MPEG-4 scene between its two formats: the Extensible MPEG-4 Textual format (XMT) [56] and the MPEG-4 BInary Format for Scenes (BIFS) [47][48]. Through the use of the MPEG-4 XMT format, MPEG-4 scenes can easily be composed and edited. Moreover, XMT has the features, which allows SMIL, VRML and MPEG-4 players to interchange information among each other. The MPEG-4 BIFS is a compressed binary format, which allows a client to have the ability to interact with a partial scene, while the rest of the scene with its details is being streamed.

A MPEG-4 player has also been designed and implemented. The player is able to display an already created MPEG-4 scene in any of the two formats. It is also able to create MPEG-4 scenes, add basic shapes to a given scene, import a scene, resize and change shapes of objects in a scene, and delete section of a scene. All this work is done on the MPEG-4 XMT format. But if a scene is in the BIFS format, it can be converted to the XMT format through the use of the developed Codec, before any editing and then converted back to BIFS (if required) when done.

Having an interesting generated MPEG-4 scene, a novel robust algorithm is developed for the authentication and protection of the MPEG-4 scene and its content. The protection algorithm is applied to the MPEG-4 XMT scenes. If the MPEG-4 scene is in the BIFS format, it can be converted to the XMT format before applying the protection algorithm or checked for authenticity.

A more general robust algorithm for the protection of SMIL (Synchronized Multimedia Integration Language) scenes is also developed. Even if the SMIL scene was modified, the algorithm can detect that given scene and locate any watermarked media objects in it.

1.5.2. Published Papers

1. Z. Sakr and N. D. Georganas, "An MPEG-4 XMT scene structure algorithm for authentication and copyright protection", Proc. IEEE 14th Inter. Workshop on Research Issues on Data Engineering, pp. 48-55, Boston, March, 2004.
2. Z. Sakr and N. D. Georganas, "A MPEG-4 XMT algorithm for scene structure authentication and content location", Proc. IEEE Canadian Conf. in Elect and Comp. Eng., vol.2, pp. 763-768, Niagara Falls, May, 2004.
3. Z. Sakr and N. D. Georganas, "Robust content-based MPEG-4 XMT scene structure authentication and multimedia content location", ACM Transactions on Multimedia Computing, Communications, and Applications, (revised and under 2nd review).
4. Z. Sakr and N. D. Georganas, "A multimedia authentication and copyright system for MPEG-4 scenes", IEEE Transactions on Signal Processing, (under review).
5. Z. Sakr and N. D. Georganas, "An authentication and copyright system for SMIL scenes", IEEE Transactions on Multimedia, (under review).

1.6. Thesis Organization

The rest of the thesis is organized as follows:

Chapter 2 gives a review of many of the watermarking algorithms proposed for watermarking different media objects. This review does not include all the watermarking algorithms developed, but provides a list of some of the most important ones developed till now.

Chapter 3 explains the designed and implemented MPEG-4 player and its features.

Chapter 4 provides a detailed explanation of the designed and implemented algorithm for MPEG-4 scene detection/authentication; and media content location and copyright protection.

Chapter 5 provides an explanation on the robust algorithm used for SMIL scenes authentication and content protection.

Chapter 6 explains the algorithm used for the protection of the media content used inside the MPEG-4 or SMIL scenes.

Chapter 7 concludes the thesis with a summary of all the research work achieved and a list of some important enhancements that can be done for future research work.

Appendix provides the reader with all the background information on the standards and techniques used and required for the understanding of the thesis research work.

Chapter 2

Watermarking Techniques

There has been a lot of research work dealing with digital watermarking in the past decade. This chapter lists and explains some of the most well known algorithms developed until now. Digital watermarking has been applied on different media files such as text, image, video, and software documents. But not all these watermarking algorithms are alike. Each has been designed for a different purpose. Some are made to be fragile and be destroyed when tampered with for authentication, while others are robust for copyright protection. This chapter also lists some of the benchmarking tools that are used for testing the watermarking algorithms on different media applications. Some standards have been put forward for digital rights management, which are also explained in this chapter. Finally, a list of some commercial watermarking software is provided.

2.1. Watermarking Types

2.1.1. Text Document Watermarking

There have been different schemes for text watermarking, which have been proposed and some in use already. Information can be hidden in the words of a text message (i.e. their meaning or order) or hidden in the text format.

Maxemchuk et al. [15], [66], [67], [74] did a lot of research on text watermarking. Information can be encoded by slight movement of objects, such as: paragraphs, lines, words, characters, figures, and captions, within the given text. For instance, a text line can be moved up to encode a '1' or down to encode a '0'. This type of watermarking can be fragile and be removed by distortions such as translation, expansion or shrinkage caused by printing, photocopying or scanning. To enforce the watermark, and make it more robust to this type of distortions, each text object that is marked (moved) is surrounded by two control objects that are not marked. For example, a marked text line is surrounded by two control lines that are not moved. This relative change in spacing between marked and unmarked objects is used for later detection of the watermark. For detection of the watermark, the centroid technique [68] can be used. It compares the difference in shift between the original text and the watermarked one for the extraction of the watermark.

Inoue et al. [46] proposed few methods for information hiding in XML, which is structured text format. For instance, in XML, end-tags can be written as "<tag></tag>" format, or as "<tag/>" format. Switching between those two formats, a "0" or a "1" bit can be embedded in the text. Moreover, tags in XML are not affected by white spaces placed in them, i.e. <tag> ⇔ <tag >. Therefore, a "0" or a "1" bit can be embedded by adding or removing white spaces in tags. There are more of such techniques presented in [44] [46].

The major drawback in all the text watermarking (hidden in the text format) proposed so far is that it can be removed easily by retyping the text using a new character font and/or removing unnecessary white spaces between text words.

2.1.2. Image Watermarking

While there has been a great amount of research on the different watermarking areas, most of the published watermarking papers are on image watermarking. The focus of all this research on image watermarking might be because of the fact that there are so many

images available on the World Wide Web free of charge and without any copyright protection.

Due to these many publications made on image watermarking, it would be hard to have a complete list of all such watermarking techniques made so far. However, most of those watermarking approaches share common principles. This review covers many of the published image watermarking papers to get the idea of all those different principles used and how they are applied in practice.

One of the first used techniques for information hiding in the spatial domain is least significant bit (LSB) embedding. These methods are relatively easy to apply in images and audio. A surprising amount of information can be hidden with little, if any, perceptible impact to the carriers. Tirkel et al. [115] [116] [118] in their early publications on watermarking explain the methods for embedding information in the LSB plane of the image data. The embedding process consists of choosing the cover data $\{c_{1\,l(i)} \ldots c_{m\,l(i)}\}$ and the watermark $\{m_1 \ldots m_n\}$, which is needed to be embedded. The LSB of a cover element $c_{l(i)}$ is then exchanged with the corresponding watermark m_i. The rest of the watermarks are embedded in the same way in their corresponding covers. In order to be able to decode the secret message, the receiver must have access to the sequence of element indices used in the embedding process. A stego-key k usable as a seed for a random number generator can be used to generate a random sequence $k_i \ldots k_n$, which can then be used as indices of the covers used for watermarking [79]:

$$c_{1\,l(i)} = k_1$$

$$c_{r\,l(i)} = c_{r-1\,l(i)} + k_r$$

The receiver at the other end needs to be informed of that secret key to be able to regenerate the indices of the covers used and recover the watermark.

A scheme explained by Kurah and McHughes [61] is known as image downgrading. Given two images of same size, one is acting as cover and the other is the secret image.

Now taking the four most significant bits of the secret image and embedding them in the four least significant bits of the cover image, the degradation of the cover image quality in most cases is unnoticeable. Extracting those four least significant bits of the cover image is enough to get a rough estimation of the secret image.

Mitsui and Tanaka [78] proposed an algorithm for embedding information using the predictive coding scheme for gray scale images. Predictive coding schemes exploit the correlation between adjacent pixels by coding the prediction error instead of coding the individual gray scale values. The image is scanned in a predefined order traversing the pixels $\{x_i\}$; i $\in$ N. The pixels are then coded using the predictive coding scheme by keeping the first value x_1 and replacing subsequent values x_i by the quantized difference Δ_i between adjacent pixels

$$\Delta_i = Q(x_i - x_{i-1})$$

To embed a watermark in a form of binary string, Δ_i is computed. If Δ_i does not match with the bit to be encoded according to the encoding table used, Δ_i is replaced with the nearest Δ_j where the associated bit equals the secret message bit. The watermark can be recovered by looking up the bit in the encoding table.

Δ_i	...	-3	-2	-1	0	1	2	3	...
	...	0	1	0	1	0	0	1	...

Table 2-1 Cipher Encoding Table

In [128], Zhao and Koch propose a watermarking method for binary images. The image is divided into blocks. Each block embeds one bit of information. Let $P_1(b)$ be the percentage of black pixels '1', and $P_0(b)$ be the percentage of white pixels '0' in a given block. A '1' bit is embedded in a given block if $P_1(b)$ is greater than a given threshold,

and a '0' is embedded if $P_1(b)$ is less than a given threshold. Some pixels in a given block might need to be changed so that the relation holds for a given embedded bit. If too many pixels must be modified to achieve the desired goal, the block is marked as invalid. A robustness degree λ against image processing is also included in the algorithm. It represents the number of bits that can be altered after image processing without damage of embedded bits.

Bender et al. [8] proposed an algorithm called "Patchwork". The algorithm simply selects pairs of pixels (a_i, b_i) randomly using a secret key K_s. The luminance values of those pairs are modified by increasing a_i by one and decreasing b_i by one. In the extraction process, the secret key K_s is used to retrieve those *N* pixel pairs. The value of the sum of the differences between the a_i's and the b_i's is used to gather some statistical properties and check whether the image has watermark embedded in it or not as follows:

$$S = \sum_{N} (a_i - b_i) = \begin{cases} 2N, for \quad watermarked \quad pairs \\ 0, for \quad nonwatermarked \quad pairs \end{cases}$$

An algorithm similar to the patchwork algorithm is proposed by Pitas and Kaskalis [90] to be used for digital signatures. Given an image *I* of size *NxM*, a binary digital signature *S* of same size as the image *NxM* with number of ones equal to number of zeros is chosen. The image *I* with luminance values x_{nm} at locations n and m is split into two subsets, by using *S*, as follows:

$$A = \{x_{nm} \in I, s_{nm} = 1\}$$

$$B = \{x_{nm} \in I, s_{nm} = 0\}$$

$$I = A \cup B$$

The watermark is superimposed by changing the elements of the subset A by a value *k*:

$$A' = \{x_{nm} \otimes k, x_{nm} \in A\}$$

The signed image is given by:

$$I_s = A' \cup B$$

The watermark is extracted using a test statistic to determine whether the watermark is present or not.

A watermarking algorithm based on inserting a bitmap logo (i.e. an image which consists of a small number of pixels of 0's and 1's) using a torus automorphism is proposed by Voyatzis and Pitas [119]. A 2D "torus automorphism" can be defined as a spatial transformation of planar regions which belong to a square 2D area. It is defined in the subset $U = [0,1)\text{x}[0,1) \subset R^2$ by the following formula:

$$\text{r'} = \text{Ar}, \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} (\text{mod} 1)$$

A dynamical system formed by iterated actions of A on a point r_0 can be expressed like a map

$$\text{r}_{n+1} = \text{A}^n \text{r}_0 (\text{mod } 1) \text{ or } \text{r}_{n+1} = \text{Ar}_n (\text{mod } 1)$$

The set of points $\{\text{r}_0, \text{r}_1, \text{r}_2, \ldots\}$ is called the orbit of the system. This system mixes the points in a chaotic way, and under certain circumstances, the automorphism may have periodic orbits, which means that after T iterations the current point is equal to the initial point, e.g. $\text{A}^T \text{r}_0 = \text{r}_0$.

A watermark is prepared by getting a square binary image and mixing it using the automorphism A_N. This mixed binary image is then added onto the original image by altering the intensity levels of the pixels in the original image. The watermark is recovered by first extracting the mixed watermark and then performing the automorphism A_{N-T}, where T is the automorphism period for the given system, to reconstruct the original watermark.

In [124], Wolfgang and Delp proposed a forgery detection scheme for image authentication (fragile watermark). A binary watermark sequence with period 2^n-1 is generated through the use of a linear feedback shift register with n stages [89], arranged into a suitable block and then added to the image pixel values, through the use of LSB modification or a similar method. This algorithm makes use of 2D watermarks, which is an extension to an algorithm done in [118]. One advantage of a 2D watermark is the

ability to more effectively locate where an image has been changed. Forgeries made to only a small portion of the image would affect the respective block and not the entire image. Let the spatial cross-correlation function of images X and Y be defined as:

$$R_{XY}(\alpha,\beta) = \sum_i \sum_j X(i,j)Y(i-\alpha, j-\beta)$$

Let X be the original image block, W be the watermark block, Y be the watermarked image block, and Z be the watermarked image block that might be forged. The test statistic for the block δ is defined as:

$$\delta = R_{yw}(0,0) - R_{zw}(0,0)$$

If the watermarked image is unchanged, δ=0. When δ is larger than a defined tolerance, the block fails the watermark test.

Kutter et al. [62] proposed a watermarking algorithm for colored images in the spatial domain that they claim to be resistant to classical attacks such as image compression and geometric attacks such as rotation, cropping and translation. Moreover, there is no need for the original image, in order to be able to extract the watermark. A single bit s is embedded into the image pseudo-randomly at position p = (i,j) using a secret key K_s. The bit is embedded by modifying the blue channel B at position p by a fraction of the luminance

L = 0.299R+0.587G+0.114B as:

$$B_{ij} \longleftarrow B_{ij}+(2s\text{-}1)L_{ij}q$$

where q is a constant determining the signature strength.

In order to recover the watermark, a prediction of the original value of pixel p containing the watermark is needed. This prediction is based on a cross-shaped neighborhood around p. The prediction $\hat{B}_{ij}$ is thus computed as follows:

$$\hat{B}_{ij} = \frac{1}{4c}\left(\sum_{k=-c}^{c} B_{i+k,j} + \sum_{k=-c}^{c} B_{i.j+k} - 2B_{ij}\right)$$

where c is the size of the cross-shaped neighborhood.

The watermarking bit can now be retrieved by comparing the predicted and the actual value of the pixel:

$$\delta = B_{ij} - \hat{B}_{ij}$$

The sign of the difference δ determines the value of the embedded bit.

This same algorithm can be applied to multiple bit embedding. Two extra bits can be added to the series of bits to be embedded. Those two bits are always a 1 and a 0 and never change. They define a geometrical reference, which is used to counter geometrical attacks, such as rotation, cropping and translation.

A novel idea in the spatial domain, which makes use of a geometric feature modification in the image to embed the watermark, instead of pixel or color modification, is described in [76]. The watermark consists of a dense line pattern, generated pseudo-randomly and superimposed over the image. The image is divided into a number of blocks and a fixed number of most significant points, called salient points, are found. Small geometric changes, called warping, are then made to the image such that a significant number of those salient points are within the vicinity of the lines. In the detection process, salient points are extracted and verified if a large number lies within the vicinity of the lines.

In [112] Tefas and Pitas proposed a watermarking algorithm, which can be used for progressive detection. The algorithm generates a three-valued watermark $w(x) \in \{0,1,2\}$, from an image f(x) given a secret key K_s. The watermark is a random sequence of three-valued data. The watermark is then embedded into the original image by altering the pixels according to the following formula:

$$f_w(x) = \begin{cases} f(x) & if \quad w(x) = 0 \\ g_1(f(x), N(x)) & if \quad w(x) = 1 \\ g_2(f(x), N(x)) & if \quad w(x) = 2 \end{cases}$$

where $N(x)$ denotes a function that depends on the neighborhood of pixels around x and

$$g_1(f(x),N(x)) = N(x) \oplus \alpha_1 f(x)$$

$$g_2(f(x),N(x)) = N(x) \oplus \alpha_2 f(x)$$

α_1 and α_2 are suitably chosen constants, that determine the watermark power.

To detect the watermark, the watermark *w(x)* is first generated using the key K_s. the detection function is defined as follows:

$$d(x) = G(f_w(x), N(x)) = \begin{cases} 1 & if \quad f_w(x) - N(x) > 0 \\ 2 & if \quad f_w(x) - N(x) < 0 \end{cases}$$

The detection function is valid if $\alpha_1 > 0$ and $\alpha_2 < 0$. Based on the watermark *w(x)* and the detection function *d(x)*, we can decide whether the watermark under investigation is embedded in the image or not.

Most of the watermarking techniques proposed so far in the spatial domain are easy to apply to an image but do not provide much resistance against image modifications, whether caused naturally such as JPEG compression or by an attacker intending to destroy the watermark. Frequency domain watermarking techniques have proved till now to be a better choice for embedding robust image watermarks.

One of the first successful watermarking algorithms in the DCT domain was proposed by Koch and Zhao [58][128]. The algorithm operates on quantized DCT coefficients and depends on the relation of three coefficients (k_1,l_1), (k_2,l_2) and (k_3,l_3) in a block to embed the watermark. One block encodes a "1", if $B_i^Q(k_1,l_1) > B_i^Q(k_3,l_3) + D$ and $B_i^Q(k_2,l_2) > B_i^Q(k_3,l_3) + D$. A block encodes a "0" if $B_i^Q(k_1,l_1) + D < B_i^Q(k_3,l_3)$ and $B_i^Q(k_2,l_2) + D < B_i^Q(k_3,l_3)$ where B_i^Q is a DCT transformed and quantized image block; and D is the minimum distance between two coefficients for representing the embedded bit. The higher D is, the more robust the method will be against image processing techniques. The blocks are then de-quantized and inversely transformed. To extract the watermark, the blocks are DCT transformed and quantized and the coefficients used to embed the watermark allocated. The rules used to embed the watermark are used again to extract it. If none of the two rules used for embedding is satisfied, then the embedded bit in that

block is damaged. The authors claim that this algorithm is robust against JPEG compression with a quality factor as low as 50%.

Bors and Pitas [14] suggest a watermarking method, which can be used for grayscale and color images and is resistant to JPEG compression. The image is first divided into 8x8 pixel blocks. Blocks situated at certain distances from each other are selected based on a Gaussian network classifier. The mid-frequency range DCT coefficients are then used for embedding. Those coefficients are modified using either a linear DCT constraints or a circular DCT detection region. Consider the linear constraint among the DCT coefficients:

$$Y = FQ$$

where F is the modified DCT coefficient vector and Q is the weighting vector provided by the watermark code. For embedding the constraint, the DCT frequency coefficients are modified based on the least squares algorithm. The second approach considers circular detection regions around certain DCT coefficients. Embedding the circular detection regions is similar to the vector quantization techniques:

$$\|F - Q_k\|^2 = \min_{i=1}^{H} \|F - Q_i\|^2 \text{ then } F = Q_k$$

where F is the DCT coefficients vector. After embedding the DCT coefficient constraints, the respective block is reconstructed, based on the inverse DCT transform. In the watermark recovery process, the algorithm first verifies the DCT coefficient constraint for all the blocks followed by a location constraint. It is claimed that the algorithm can accommodate JPEG compression ratios of 13:1 and 18:1 using the linear DCT constraint or the circular detection region constraint respectively.

Tao and Dickensn [111] introduced an adaptive watermarking technique using the DCT domain to embed the watermark. A regional classifier is employed to assign a noise sensitivity index to each region. The watermark is inserted into the image according to the index using block DCT. The watermark is added to the N AC coefficients having the

smallest quantization step sizes according to the default JPEG compression table. The following formula is used to add a signal at the *i*th AC position in a block:

$$\hat{x}(i) = x(i) + max[x(i)*\alpha_m, sgn(x(i))*D_i/\kappa]$$

where m is the noise sensitivity index for the block, $x(i)$ is the ith DCT coefficient with $\hat{x}(i)$ being its watermarked counterpart, D_i is the ith quantization step size in the JPEG table and κ satisfies $5 \leq \kappa \leq 6$. Note that the watermark signal is not generated randomly. A noise sensitivity index is assigned to each block, which makes use of the masking effects of the HVS, both to protect regions sensitive to noise and insert strong signals in regions with low noise sensitivity. In the classification algorithm, properties such as luminance masking, edge masking and texture masking effects are exploited. It classifies a block into one of six perceptual classes, from 1 to 6, edges, uniform with moderate intensity, uniform with either high or low intensity, moderately busy, busy, and very busy, in descending order of noise sensitivity. Each perceptual class has a noise sensitivity index assigned to it. For the watermark extraction, the original image, as well as the watermark, are required. Experimental results reveal that the method resists JPEG compression down to a quality of 5% and can accommodate random noise with a peak signal-to-noise ratio (PSNR) of 22.1dB.

Cox et al. [21] made use of the idea of spread spectrum communication to spread the watermark over frequency components of an image. From an image, a sequence of values $V = v_1, \ldots, v_n$, are extracted into which a watermark $X = x_1, \ldots, x_n$ is inserted to obtain an adjusted sequence of values $V' = v_1', \ldots, v_n'$ and then inserted back into the image. A scalar parameter α is introduced, which determines the extent to which X alters V using one of the following three formulas:

$$v_i' = v_i + \alpha x_i \qquad (1)$$

$$v_i' = v_i(1 + \alpha x_i) \qquad (2)$$

$$v_i' = v_i(e^{\alpha x_i}) \qquad (3)$$

Using this algorithm, equation (2) is mostly used. A single scaling parameter α may not be applicable for perturbing all of the values v_i, since different spectral components may

exhibit more or less tolerance to modification. Using multiple scaling parameters $\alpha_1,\ldots,\alpha_n$ and using update rules such as $v_i' = v_i(1 + \alpha_i x_i)$ might be more appropriate. To verify the presence of the watermark, the similarity between the extracted watermark X^* and the original watermark is measured as follows:

$$\text{sim}(X, X^*) = \frac{X^*.X}{\sqrt{X^*.X}}$$

Experimental results showed that the method resists JPEG compression at a quality factor of 5%, scaling, dithering, cropping, printing/scanning, and collusion attacks.

Podilchuk and Zeng [92] [93] [94] introduced an image adaptive watermarking algorithm. Through the use of image models, the watermark can achieve maximum length and maximum power when embedded into the image. The models used in the algorithm can be described in terms of three different properties of the human visual system that have been studied in the context of image coding: frequency sensitivity, luminance sensitivity and contrast masking [122] [102]. Two watermarking schemes were proposed: image-adaptive DCT (IA-DCT) and image-adaptive wavelet (IA-W) schemes. The watermark insertion in both schemes can be described in general as:

$$X^*_{u,v} = \begin{cases} X_{u,v} + J_{u,v} w_{u,v} & \text{if} \quad X_{u,v} > J_{u,v} \\ X_{u,v} & \text{otherwise} \end{cases}$$

where $X_{u,v}$ refers to the frequency coefficients of the original image samples $x_{i,j}$, $X^*_{u,v}$ refers to the watermarked image coefficients, $w_{u,v}$ is the sequence of watermark values generated and $J_{u,v}$ is the computed just noticeable difference (JND) calculated for each coefficient. In the IA-DCT, Watson's model [122], which is based on frequency sensitivity, luminance sensitivity and contrast masking components, is used for determining the JNDs. For the watermark detection, the original image is subtracted from the received image and the correlation between the signal difference and a specific watermark sequence is determined. The correlation value is compared to a threshold to determine whether the received image contains the watermark in question. Experiments

showed that the watermark is extremely robust to JPEG compression, cropping scaling, additive noise, printing/scanning. For attacks involving a geometrical transformation, the inverse operation has to be applied to the image before the watermark detection process.

Ruanaidh et al. [101] introduced the use of Mellin-Fourier transform for watermarking. The transform space of Mellin-Fourier is based on the translation property of the Fourier transform:

$$F(k_1,k_2)exp[-j(ak_1+bk_2)] \leftrightarrow f(x_1+a, x_2+b)$$

Note that shifts in the spatial domain cause a linear shift in the phase component. As a consequence, the workspace in which the watermark will be embedded is limited to the subspace related to the amplitude of the Fourier transform, which will be insensitive to a spatial shift in the image. The basic translation invariants may be converted to rotation and scale invariant by means of a log-polar mapping (LPM) defined as follows:

$$(x,y) \rightarrow \begin{cases} x = e^{u}\cos\vartheta & with \quad u \in \Re \quad and \quad \vartheta \in [0,2\pi] \\ y = e^{u}\sin\vartheta & with \quad u \in \Re \quad and \quad \vartheta \in [0,2\pi] \end{cases}$$

Note that rotation and scale in the Cartesian system will result in a translation in the logarithmic coordinate system. The property of translation invariance can thus be used to construct a space insensitive to any rotation or scale operations carried out on a watermarked image. This algorithm was tested using spread spectrum encoding using Reed Solomon (RS) invariant domain embedding. The image was rotated by 143^{o} and scaled by a factor of 75% along each axis. The watermark was detected and extracted successfully from this test.

In [129][130], Zheng et al. proposed a watermarking algorithm that is invariant to rotation, scaling, and translation (RST). The algorithm is based on the log-polar mapping (LPM) and phase correlation. The watermark is embedded in the LPMs of the Fourier magnitude spectrum of an original image. The phase correlation between the LPM of the original image and the LPM of the watermarked image is used to calculate the displacement of watermark positions in the LPM domain. It is said that the scheme

preserves the image quality by avoiding computing the inverse log-polar mapping (ILPM), and produces smaller correlation coefficient for unwatermarked images by using phase correlation to avoid exhaustive search. The experiments proved the algorithm to be robust against rotation, translation, and scaling transformations within reasonable limits, and JPEG compression.

There has also been some considerable research work on watermarking in the wavelet domain. Wavelets are becoming a key technique in the ongoing source compression standard JPEG2000. Watermarking is used in the wavelet domain to resist compression in JPEG2000, the same as watermarking in the DCT domain was used to resist JPEG/MPEG compression.

Xia et al. [125] proposed a watermarking scheme in the wavelet domain. In the encoding part, the image is decomposed into several bands and a pseudo-random sequence (Gaussian noise) is then added to the large coefficients, which are not located in the lowest resolution (the left top corner; see Figure 2-1). The Gaussian noise N(m,n) with mean 0 and variance 1 is added to the DWT coefficients y(m,n) not located at the lowest frequency band of image x(n,m) as follows:

$$\tilde{y}(m,n) = y(m,n) + \alpha [y(m,n)]^2 N(m,n)$$

where α controls the level of the watermark and the square power indicates the amplification of the large DWT coefficients. The watermarked image is then obtained by taking the IDWT of the modified DWT coefficients $\tilde{y}$ with the unchanged DWT coefficients at the lowest resolution. For watermark detection, given the original image and the received image, both images are decomposed with DWT into four bands: low-low (LL_1), low-high (LH_1), high-low (HL_1), and high-high (HH_1) bands. The signature added in the HH_1 band is then compared with the difference of the DWT coefficients in the HH_1 bands of the received and the original images by calculating their cross correlation. If there is a peak in the cross correlation, the signature is detected. Otherwise, the signatures added in the HH_1 and LH_1 bands are compared with the difference of the

DWT coefficients in the HH_1 and LH_1 bands respectively. If a peak was formed, then the watermark is detected. Otherwise the signatures added in the HL_1, LH_1 and HH_1 bands are considered. If there is still no peak in the cross correlation, the original and the received images are decomposed further in the LL_1 band into four additional subbands LL_2, LH_2, HL_2, and HH_2 and so on until a peak in the cross correlation appears. Otherwise, the watermark is not detected. The scheme was tested under additive noise, compression, and halftoning and has been claimed to show some satisfactory results.

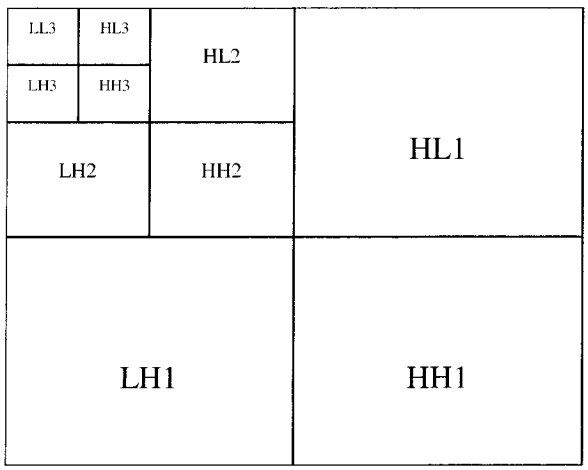


Figure 2-1 DWT pyramid decomposition of an image

Kundur and Hatzinakos [60] introduced an algorithm in which, through the use of a human visual system (HVS) model, a multiresolution data fusion approach is used for embedding, where the image and the watermark are both transformed into the discrete wavelet domain. The algorithm is composed of three steps for embedding a watermark. First, the host image and the watermark are transformed into the wavelet domain. The Lth level discrete wavelet decomposition is performed to produce a sequence of $3L$ detail images, corresponding to the horizontal, vertical and diagonal details at each of the L resolution levels and a gross approximation of the image at the coarsest resolution level. The kth detail image component at the ith resolution level of the host is denoted as $f_{k,l}(m,n)$ where k = 1,2,3 and l = 1,..., L. For the watermark, only the first level discrete

wavelet decomposition is performed. The resulting N_{wx} x N_{wy} coefficients can be denoted as $w_{k,l}(m,n)$. After the host image and the watermark have been converted to the wavelet domain, the detail images of the host at each resolution level are segmented onto non-overlapping N_{wx} x N_{wy} rectangles. The segments are denoted by $f^i_{k,l}(m,n)$. The salience $S(x)$, which measures the importance of an image component, of each of these localized segments is computed using information about the HVS. The watermark is embedded by a simple scaled addition of the watermark to the particular N_{wx} x N_{wy} detail component as follows:

$$g^i_{k,l}(m,n) = f^i_{k,l}(m,n) + \gamma_{k,l}\sqrt{S(f^i_{k,l}(m,n))}w_{k,l}(m,n)$$

The user defined parameters $\gamma_{k,l}$, for l = 1,..., L, are positive real numbers which determine a trade off between visibility of the watermark and its robustness to signal distortions at each of the resolution levels. The third and last step for this algorithm consists of performing the Lth level inverse wavelet reconstruction of the fused image components to form a watermarked image. The watermark is extracted form the possibly corrupted watermark image using the host image, by applying the inverse procedure at each resolution level to obtain an estimate of the watermark. The estimates for each resolution level are averaged to produce an overall estimate of the watermark. The watermark was tested for JPEG compression, additive noise and two-dimensional linear mean filtering. The watermark was extracted after performing compression around 34, SNR of 15dB and filtered using a KxK linear mean filter with $K \geq 4$.

Guo and Georganas [35] proposed a joint ownership watermarking algorithm in the wavelet domain. The algorithm makes use of Shamir's threshold scheme (used in cryptography applications) to generate several private keys. No single owner can claim the possession of a watermarked image without the presence of the other owners. Combining the keys of the owners will form the main key used to generate the watermark. The watermark is embedded into the image through a scheme proposed by Cox et al. [21].

Lu et al. [69] proposed a watermarking scheme, called "Cocktail watermarking" in the wavelet domain, which embeds two complementary watermarks in a host image so that at least one watermark survives under different attacks. This scheme makes use of the ideas of spread spectrum watermarking in [21] and of image adaptive watermarking based on [94]. An arbitrary attack usually tends to increase or decrease the magnitudes of the majority of the transformed coefficients. That is, the chance of an attack of making the number of increased and the number of decreased coefficients equal is very low. The behavior of attacks can be classified into two; the first category contains attacks like compression and blurring, which tend to decrease the magnitudes of most of the transformed coefficients of a watermarked image. By using a modulation of positive quantity to negative coefficient Modu(+, -) or a negative quantity to a positive coefficient Modu(-, +) should contribute positively to the detector response. On the other hand, the second attacks contain those attacks such as sharpening and histogram equalization, which have the tendency to increase most of the magnitudes of transformed coefficients, then every constituent transformed coefficient should be modulated with a quantity that has the same sign. Under these circumstances, only Modu(+, +) and Modu(-, -) will contribute positively to the detector response. A good modulation strategy should take the behavior of attacks into account. The region used to hide the watermarks is divided into two parts, the lowest frequency part and a part that covers the remaining frequencies Hence, different weights may be assigned to achieve a compromise between transparency and robustness. Only the frequency masking effect of the wavelet-based visual model is considered. A set of wavelet coefficients is selected if their magnitudes are larger than their corresponding JND thresholds. Two watermarks, which play complementary roles in resisting various kinds of attacks, are embedded into the image. Because two complementary watermarks used to be hidden, the length of each watermark should be one half the amount of the total of the selected coefficients. The watermark design is image-adaptive [94]. The two hidden watermarks are also spread randomly in the wavelet domain. For detection purposes, the original image is needed to extract the watermark. The watermarking scheme was tested under different cases and shown that even in some

cases the watermark is severely affected by the attack at least one of the watermarks still survived this attack.

There has also been some work on image watermarking using fractal image coding. The fractals theory has proved to be suitable in many fields and particularly in various applications of image compression. The main idea is that, given an image O we want to encode, let O_r denote the partition of O in nxn blocks be referred to as Range blocks (Rb) and O_d the partition of O into 2nx2n blocks in steps of nxn pixels as Domain blocks (Db). The goal of the encoding algorithm is to establish a relationship between O_r and O_d in such a way that any Rb can be expressed as a set of transformations (contraction, isometric transformation, luminance scaling, and luminance shifting) to be applied on a particular Db. For each Rb in O, denoted as Rb_j, the code will consist of a vector V_j and the appropriate transformations T_j, in such a way that [97]:

- V_j has its origin in Rb_j and points to the correspondent Db_j which now becomes its Matching block (MB_j).
- T_j if applied to Mb_j, minimizes the Mean Square Error (MSE) with respect to Rb_j.
- The couple { V_j, T_j } is the best solution (in the sense of the MSE) within a local area surrounding Rb_j in which a search is made for Mb_j.

The region of O_d where the search of Mb_j is performed is commonly taken as a square region surrounding the Rb_j known as LSR (local searching region).

For watermarking purposes, as proposed by Puate and Jordan [97], given two different LSR, A and B, and a third one, C, defined as their union as shown in Figure 2-2 and having a watermark S = {s_0 , ..., s_n }with every bit embedded with a redundancy U, the embedding process is defined as follows:

- For each bit s_i, U Range blocks are randomly chosen and denoted by $\{Rb\}_i$. The random function used to generate the blocks is based on a seed which is known only to the user.
- If $s_i = 1$, $\{Rb\}_i$ is coded by searching for $\{Mb\}_i$ in regions $\{A\}_i$
- If $s_i = 0$, $\{Rb\}_i$ is coded by searching for $\{Mb\}_i$ in regions $\{B\}_i$
- The rest of Rb_j are coded by searching for Mb_j in $\{C\}_i$.

The watermark can be extracted by simply accessing the Range blocks of image Q defined by the 'seed' used when signing, recoding them and checking the values of V_j.

It can be decided if a Range block has been signed with a '1' or '0' as follows:

- If V_j belongs to region A_j, then a '1' has been embedded.
- If V_j belongs to region B_j, then a '0' has been embedded.

The algorithm was tested against JPEG compression and low pass filtering. The algorithm was able to retrieve the watermark up to JPEG quality of 75% with a reliability of 40.8%. But for low pass filtering, the algorithm showed weakness against blurring convolutions.

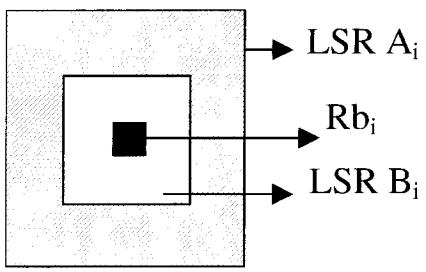


Figure 2-2 Watermark searching region

Bas et al. [6] proposed a watermarking scheme using a fractal code both in the spatial domain and in the DCT domain. The algorithm adds artificial and visually invisible local similarities into the image by substituting a Range block with a new block $\hat{R}$.

- In spatial domain: For each Domain block D and Range block R, $\hat{R}$ is calculated as follows:

$$\hat{R} = \delta * S * \frac{D}{\max(D)} + \bar{R}$$

where $\bar{R}$ is the mean of R and the magnitude of the watermark is fixed by a factor S.

$$\delta = \begin{cases} +1 & if \quad the \quad embedded \quad bit = 1 \\ -1 & if \quad the \quad embedded \quad bit = 0 \end{cases}$$

The quadratic error between R and $\hat{R}$ is then calculated.

- DCT domain embedding: The 8x8 blocks are transformed to DCT coefficients. $\hat{R}$ is modified as follows:

$$\hat{R} = R_{hp} + \delta * S * \frac{D_{lp}}{\max(D_{lp})}$$

where .hp and .lp are the high-pass components (plus the DC coefficient) and the low-pass components respectively.

$$\delta = \begin{cases} +1 & if \quad the \quad embedded \quad bit = 1 \\ -1 & if \quad the \quad embedded \quad bit = 0 \end{cases}$$

Similarities are searched in the luminance space to avoid quantization problems after the inverse DCT transform.

For watermark extraction, a domain block is obtained. If the index R is the same as the index in the table, p_1 (counter for matched blocks) is increased and the embedded bit is deducted from the sign of δ. If the index R is not the same as the index in the table, p_1 does not change. Another Domain block is then obtained and the search starts again until no more D blocks. For the same distortion (PSNR = 52dB), the DCT scheme has been proven to be more robust to JPEG compression than the spatial domain scheme. Moreover, it has been claimed that if the image does not contain edges or contains low-dynamic edges, the detection step is less robust to compression techniques.

This section presented different image watermarking methods. The watermarking methods are based on the same principle: pseudorandom changes are applied to selected coefficients in the spatial or transform domain. Those changes are later identified by

correlation or correlation-like similarity measures. As was explained, those different watermarking methods have their strengths and weaknesses. Watermarking in the spatial domain lacks robustness to lossy JPEG like compression, but is easy to recover if the image was cropped or translated. Cropping in the frequency domain on the other hand results in substantially large distortion, which usually destroys the watermark. But watermark embedding in the frequency domain is usually more robust to noise like distortions than watermark embedding in the spatial domain. Depending on the type of attacks, the watermark will go through and embedding/extraction speed required, a watermarking method may be selected.

2.1.3. Video Watermarking

All the watermarking methods applied on an image can be applied on video. But video watermarking has some other limitations to consider. For instance, a video might be in a compressed format. To decompress a video, watermark it and then re-compress it, might not be feasible/practical to be done in real time. Therefore watermarking in the compressed domain might be required. Moreover, a video watermark should resist different types of attacks, such as frame averaging, frame dropping and frame swapping by distributing watermark information over several consecutive frames. In the following, different watermarking techniques for video in the compressed and uncompressed domain are discussed.

Hartung and Girod [39] proposed two watermarking algorithms, one for uncompressed video and the other for compressed video. The algorithm for watermarking in the uncompressed domain makes use of spread spectrum communications. The embedding process is the same as the algorithm proposed by Cox et al. for image watermarking [21]. The difference is in the extraction process. Knowing that the pseudo-noise signal p_i, used for watermarking, should be kept private to avoid attacks on the watermark, a public pseudo-noise signal p_i^{public} is constructed to allow users to detect a watermark presence without the privilege of destroying it. The public pseudo-noise signal p_i^{public} is

constructed by taking on average each nth coefficient for the original pseudo-noise sequence and all other coefficients are arbitrary random values with the same distribution as the pseudo-noise signal p_i.

$$p_i^{public} = \begin{cases} p_i & probability \quad of \quad (1/n) \\ rand\{-1,+1\} & else \end{cases}$$

The watermark can now be detected by applying the summation for each watermark bit:

$$s_j^{public} = \sum_{i=j.cr}^{(j+1).cr-1} p_i^{public} . \bar{v}_i$$

where $\bar{v}_i$ is the filtered version of the watermarked video signal.

The watermark bit can be recovered by thresholding:

$$\hat{a}_j^{public} = sign(s_j^{public})$$

Attacks using the public pseudo-noise signal are possible to tamper with the watermark. The value of the correlation sum for the 'whole' watermark when decoding with the secret pseudo-noise signal p_i can be decreased but not destroyed.

The second algorithm for watermarking of compressed video proposed by Hurtang and Girod [37] is embedded in the MPEG-2 bitstream domain. The incoming MPEG-2 bitstream is split into header and side information, motion vectors and DCT encoded signal blocks. Only the DCT encoded signal blocks are altered in the watermarking embedding process. The DCT encoded signal blocks are represented by a sequence of Huffman codes. Each incoming Huffman code is decoded (EC^{-1}) and inversely quantized (Q^{-1}). A quantized DCT coefficient of the current signal block is then obtained. The corresponding DCT coefficient from the transformed watermark block is then added yielding a watermarked DCT coefficient (see Figure 2-3). The watermarked coefficient is then quantized (Q) and Huffman encoded (EC). The number n_1 of bits of the new Huffman code word is then compared to the number n_0 of bits of the old unwatermarked coefficient. The watermarked coefficient is transmitted if $n_1 \leq n_0$. Otherwise, the unwatermarked DCT coefficient is transmitted and the watermarmark cannot be

embedded into that DCT coefficient. No details on robustness have been mentioned on this watermarking algorithm, but it has been claimed to be robust against filtering and quantization in the pixel and the frequency domain.

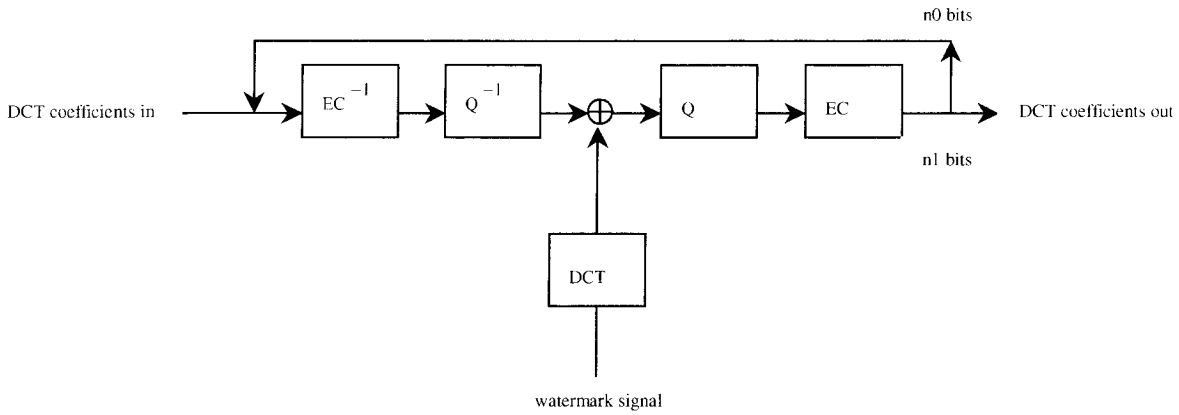


Figure 2-3 Watermarking of compressed video [37]

Langelaar et al. [64] introduced real-time watermarking algorithms for compressed video. The first algorithm is defined as "labeling in the bit domain". The label L consisting of bits L_i(i = 0,1,2,...,n) is embedded in the MPEG stream by selecting suitable variable-length codes (VLC) and forcing the least significant bit (LSB) of their quantized level to the value of L_i. To ensure that the change in the VLC yields perceptually invisible degradations after decoding and that the MPEG bitstream keeps its original size, only those VLC's (label bit carrying VLC "lc-VLC") for which another VLC exists with the same run length, a level difference of 1 and the same codeword length are selected. To add the label bitstream L to an MPEG video bitstream, the VLC's in each macroblock are tested. If an lc-VLC is found and the least significant bit of its level is unequal to the label bit L_i, then that VLC is replaced by another, whose LSB level represents the label bit. If the LSB of its level equals the label bit L_i , the VLC is not changed. This procedure is repeated until all label bits are embedded. To extract the label bits, the VLC's in the macroblocks are tested. If a lc-VLC is found, the value represented by its LSB is assigned to the label bit L_i. The procedure is repeated for all i = 0,1,2,..n until no more lc-VLC

can be found. The easiest way to destroy the watermark using this algorithm can be done by decoding the labeled MPEG stream and encoding it again using another bit-rate.

The second algorithm "coefficient domain labeling concept" introduced by Langelaar et al. [64] is claimed to be more robust against attacks than the first algorithm even though it is more computationally demanding. This algorithm embeds the label bitstream L consisting of bits L_i (i = 0,1,2,...n) in the I-frames only. Each bit out of the label string has its own bit carrying region lc-region in an I-frame. For example, the first labeling bit is located in the top left corner of the I frame, in an lc-region of n=16 8x8 pixel blocks as shown in Figure 2-4.

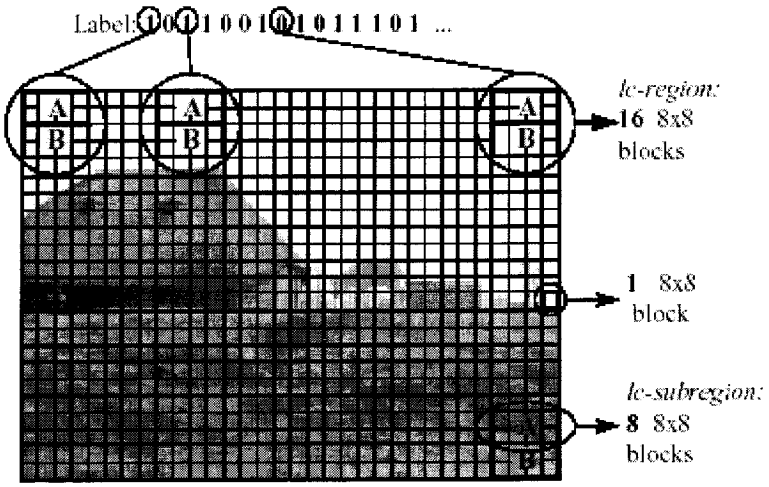


Figure 2-4 Label bit positions and block definitions in a frame [64]

A label bit is embedded in an lc-region by introducing an energy difference between high frequency DCT coefficients of the top half of the lc-region (lc-subregion A) and the bottom half (lc-subregion B). The energy E_A for subregion A is calculated as follows:

$$E_A(c) = \sum_{b=0}^{n/2-1} \sum_{u \in S(c)} (DCTcoeff_{(u,b)})^2$$

where u represents the DCT coefficient number after the zig-zag scanning, b represents the DCT block number in lc-subregion A and $S(c)$ is defined according to a cut-off point in the zig-zag scanned DCT coefficients.

$$S(c) = \{u \in [0,63] \mid (u > c)\}$$

The energy can be calculated in the same way for subregion B.

The energy difference D between top and bottom half of an lc-region is defined as:

$$D = E_A - E_B$$

A label bit "0" is defined as $D > 0$ and a label bit "1" as $D < 0$.

Now to embed a watermark, the energies E_A and E_B have to be manipulated to satisfy the requirements. To embed a "0", all the energy after the cut-off point in the DCT blocks of lc-subregion B is eliminated by setting the corresponding DCT coefficients to zero. To embed a "1", all the energy after the cut-off point in the DCT blocks of lc-subregion A is eliminated. To extract the watermark, the cut-off point has to be determined through thresholding. The selection of a suitable cut-off point for an lc-region is very important for the robustness and the visibility of the label bit. This proposed algorithm is claimed to be more robust than the previous one (labeling in the bit domain), but slightly more computationally demanding. Removing the label by labeling the stream again using another label and other pseudo random block shuffling is not possible. The watermark was tested for bit errors after transcoding a watermarked 8Mbits/sec MPEG-2 sequence at lower bit rate. It has been claimed that if 50% bit errors were made, the watermark is completely removed. From the experiments, if the video bit rate is decreased by 38%, 21% watermark bit errors were introduced and the watermark will still be present.

Dittmann et al.[29] applies two image watermarking algorithms [58] [32] to video. The first watermarking scheme makes use of the Koch-Zhao proposed algorithm [58] with a simple enhancement using the human visual system (HVS) characteristic. The strength of the embedded watermark adapts to the HVS properties by simply using the smoothness and edge characteristics of a DCT block as follows:

$$\text{Level} = \text{smoothscale}*\text{smooth} + \text{edgescale}*\text{edge} + \text{offset}$$

where smooth is the number of non-zero DCT coefficients after quantization (high values indicate great visual tolerance against additional distortion through the watermark), edge is the sum of the absolute values of the DCT coefficients 1,2,8,9,10,16,17 (high values in

these coefficients indicate that the block could have edge characteristics) and the rest of the values (smoothscale, edgescale and offset) used to change watermark strength are set to a constant value through experimentation. The quantized DCT block is calculated based on the Level value obtained for that block before processing and the watermark is then embedded into that block using three coefficients, as proposed in [58].
The second algorithm is applied in the spatial domain making use of the Fridrich algorithm [32]. A k value (determined from the Level value calculated for that block) is added/subtracted to the luminance value of the watermarked block.
Both algorithms were tested for robustness using StirMark. Using the Koch-Zhao algorithm [58], StirMark was able to remove 30% of the watermark using an error correcting code and destroy it completely without it. The results obtained from Fridrich's algorithm [32] were even worse even with the use of the error correction.

In [17], Busch et al. applies to video the method proposed in [58] for still image watermarking on DCT blocks. The watermarks are embedded into the luminance components of uncompressed video and retrieved after decompression. Edge detection and plain area detection mechanisms are employed to improve the invisibility of the watermarks. The experimental results reveal that a bit error between 0-50% is caused when extracting a watermark after a 64bit label is embedded into each frame of an ITU-R 601 video followed by MPEG-2 compression at 4-6Mbits/sec. The authors propose to embed the watermark into several frames and average them out on retrieval to increase the accuracy of the extracted watermark.

A novel multiresolution video watermarking scheme has been proposed by Swanson et al. [110]. This watermarking technique takes into account attacks such as frame averaging, frame dropping, frame swapping, collusion and statistical analysis. The watermarking scheme makes use of image masking models based on HVS to ensure that the watermark embedded into each video frame is perceptually invisible and robust. Moreover, applying an identical watermark to each frame in the video leads to problems of maintaining statistical invisibility and applying independent watermarks to each frame also is a

problem, since successive video frames may be statistically compared or averaged to remove independent watermarks. Those problems have been considered in this algorithm. A wavelet transform applied along the temporal axis of the video results in a multiresolution temporal representation. In particular, the representation consists of temporal low-pass frames and high-pass frames. The low-pass frames consist of the static components in the video scene and the high-pass frames capture the motion components and the changing nature of the video sequence. The watermark is embedded into those components. The watermark embedded in the low-pass frames exists throughout the entire video scene. The watermark embedded in the motion frames is highly localized in time and changes rapidly from frame to frame. Using this technique, frame averaging will damage the dynamic watermark only and not the static one. The watermark is embedded in the video as follows:

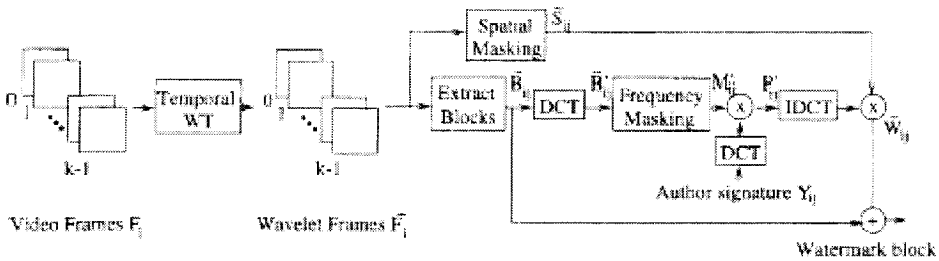


Figure 2-5 Video watermarking scheme [110]

Each wavelet frame $\tilde{F}_i$ is segmented into 8x8 blocks $\tilde{B}_{ij}$ as shown in Figure 2-5. The DCT block $\tilde{B}_{ij}$' of the frame block $\tilde{B}_{ij}$ is computed and the frequency mask M'_{ij} of the DCT block $\tilde{B}_{ij}$' is calculated. Using the author's signature Y_{ij}, the frequency shaped author signature $P'_{ij} = M'_{ij} Y_{ij}$ is obtained. The wavelet coefficient watermark block $\tilde{W}_{ij}$ is calculated by computing the inverse DCT of P'_{ij} and the watermark is increased to the maximum tolerable error level by using the spatial mask $\tilde{S}_{ij}$. The watermark $\tilde{W}_{ij}$ is added to the block $\tilde{B}_{ij}$ creating the watermarked block. The process is repeated for each wavelet

coefficient frame $\tilde{F}_i$. The watermark detection is done by hypothesis testing, detecting whether the watermark is there or not. Experimental results show the robustness of the scheme against additive noise, MPEG video compression, and frame dropping. A disadvantage of the scheme is that it has a very high complexity including model of the HVS and a blockwise DCT.

Deguillaume et al. [27] proposed a video watermarking algorithm. The algorithm works with uncompressed video. It is supposed to resist to frame cropping and padding, video frame re-sampling, aspect-ratio modification and MPEG compression. The watermarking algorithm does not need the original video to extract the watermark. The algorithm embeds a spread-spectrum watermark into blocks of video by employing a 3D DFT and adding to the transform coefficients. However, the watermark is not robust as is to frame cropping, frame scaling, frame rotations and frame-rate. This is because these transformations modify the position of the embedded signal inside the DFT magnitude and spread-spectrum sequences need perfect synchronization to work properly. Retrieving these new positions is equivalent to recovering the geometrical transformation applied to the video. To be able to estimate the parameters of these transformations, a template is inserted into the 3D DFT magnitude in addition to the watermark. To simplify the search of the transformed template in the modified video, a log-type mapping was used which allows to transform the scaling or rotation into simple shift operations. From the experimental results, it has been shown that the algorithm is robust against MPEG compression, but the error grows up to around 20% in the presence of aspect-ratio changes and frame-rate changes, even though the changes are recognized with the help of the template inserted.

Research work was also in the MIRADOR [82] project to upgrade the existing watermarking techniques developed within the MPEG-2 framework, to the new issues arising within the MPEG-4 standard. MIRADOR aims at integrating MPEG-2 watermarking technologies into the MPEG-4 for both video and audio by optimizing the

techniques to the MPEG-4 constraints. The project is intended not only to be innovative with the watermarking algorithms, but also to work closely with the standards body to ensure that watermarking is integrated and recognized as a key enabling technology for content protection of MPEG-4 objects. Consequently, the project has an important objective to analyze and actively participate to the MPEG-4 ad hoc working groups so that the technology is accepted and integrated at the level of the MPEG-4 system and that associated hooks for coupling watermarking and monitoring (capability of reading back watermarks) are specified.

Piva et al. [91] proposed a robust watermarking algorithm for the protection of MPEG-4 video. The MPEG-4 coded video bit-stream is decoded to obtain a sequence of frames. Every object is then extracted from every frame of the video sequence as an individual image. The method applies the discrete wavelet transform (DWT) to the whole extracted object image and the watermark is embedded into the wavelet coefficients belonging to the three detail bands at level 0. After every extracted image is marked, they are mixed together to obtain the frame containing the copyright information of the objects present. It has been shown that even if a video object has been transferred from one sequence to another, the copyright data of the single object can still be detected correctly.

In [5], Barni et al. proposed a MPEG-4 watermarking algorithm. The algorithm embeds a watermark in each video object of a MPEG-4 coded video bit-stream by imposing specific relationships between some predefined pairs of quantized DCT middle frequency coefficients in the luminance blocks of pseudo-randomly selected macro blocks. The quantized coefficients are recovered from the MPEG-4 bit-stream, modified to embed the watermark and then encoded again.

2.1.4. Audio Watermarking

Some of the schemes proposed for image and video watermarking can also be used for audio watermarking. But a different problem has to be considered before watermarking

audio. The human audible system (HAS) is much more sensitive than the HVS and this has to be taken into consideration to have an inaudible watermark embedded into an audio file.

Boney et al. [13] proposed a similar technique to [21] using spread spectrum for audio watermarking. This algorithm makes use of the HAS to embed the watermark. A PN sequence is first generated. A masking threshold of the audio signal is calculated using the MPEG Audio Psychoacoustic Model. The masking threshold is determined on consecutive audio segments of 512 samples. The PN sequence is then filtered with an approximate masking filter in order to ensure that the spectrum of the watermark is below the masking threshold. Since the spectral content of the audio signal changes with time, watermarks added to different blocks will be in general different even if they are generated from the same starting PN-sequence. For the detection of the watermark, it is assumed that the authorized user has access to the original signal and the PN-sequence used for watermarking that signal. The detection method is based on hypothesis testing to check whether the watermark is present or not. The technique used shows robustness to MPEG-1 audio layer III coding.

Arnold and Kanka [1] proposed an audio watermarking scheme, which has been adapted to the frequency domain and does not require the original in order to detect the watermark. This scheme makes use of a statistical method called the patchwork algorithm [8]. Given a data set containing 2N values (frequency coefficients of the Fourier domain), two intermixed subsets $A = \{a_i\}_{i=1,\ldots,M}$ and $B = \{b_i\}_{i=1,\ldots,M}$ of equal size $M \leq N$ are pseudorandomly selected from the original set. The selected elements $a_i \in A$ and $b_i \in B$, $i = 1,\ldots, M$ are altered according to the embedding functions e_A and e_B respectively. This watermarking technique uses hypothesis testing to check whether the watermark is present or not. It is claimed that the algorithm is robust against MPEG-1 Layer III compression, low and high pass filtering, resampling, requantization and cropping.

Qiao and Nahrstedt [98] introduce two watermarking schemes, which embed the watermark directly into the MPEG audio bit streams. One embeds the watermark into the Scale Factors of the MPEG audio streams and the other one embeds the watermark into the MPEG encoded samples. To generate the watermark, a key K is selected and for each MPEG audio frame a_j, j = 1, ..., N, the encryption algorithm DES is applied with key K to get a random byte sequence RBS

$$\text{RBS} = \text{DES}_K \text{ (one audio frame } a_j)$$

Let RBS_i be the i-th byte of the random byte sequence and w_i be the i-th bit of the watermark bit stream, then the watermark is generated as follows:

$$w_i = \begin{cases} -1 & if \quad RBS_i = even \quad number \\ 1 & otherwise \end{cases}$$

It has been claimed that a small change of scale factor level (i.e. increase or decrease by 1) will not be noticeable by the listener. Each scale factor takes 6 bits, therefore there are as many as 63 levels of scale factors (indexed from 0 to 62, 63 is not used by the standard). The watermark embedding algorithm is very simple and is described as follows:

$$\text{ScaleFactorW}_i = \begin{cases} ScaleFactor_i(index) & if \quad index + w_i = -1 or 63 \\ ScaleFactor_i(index + w_i) & otherwise \end{cases}$$

where $\text{ScaleFactor}_i(\text{index})$ is the i-th scale factor with the level indicated by index and ScaleFactorW_i is the i-th watermarked scale factor.

The drawbacks of this scheme though is that in some audio streams there are only few scale factors for a frame, and therefore this watermarking scheme does not have much data to watermark. Moreover, using this scheme multiple watermarks cannot be applied since increasing the scale factor by 2 levels or more will often create a perceivable audio distortion and the noise can be heard.

The second proposed scheme embeds the watermark into the sample data. But from tests being done, it has been shown that by adding 1 or –1 to every encoded sample, the distortion of the resulting audio is easily detected by the human ear. This problem was

solved by introducing a spacing parameter sp and for every sp samples, 1 or 2 samples are randomly selected for watermarking. The watermark is created as follows:

$$w_i = \begin{cases} -1 & if \quad RBS_i = 0 \pmod{sp} \\ 1 & if \quad RBS_i = 1 \pmod{sp} \\ 0 & otherwise \end{cases}$$

The watermarking procedure is done as follows:

$$\text{SampleW}_i = \begin{cases} Sample_i & if \quad every \quad bit \quad of \quad (Sample_i + w_i) \quad is \quad 1 \\ Sample_i + w_i & otherwise \end{cases}$$

where Sample_i is the ith sample in the audio frame and SampleW_i is the ith watermarked sample. It has been shown that by choosing a good spacing parameter (depending on different audio streams), the distortion can be minimized.

Dittmann et al. [30] proposed a watermarking scheme where the watermark is inserted into the MPEG bit stream. The scheme uses the same idea as [98] of modifying the scale factors of the MPEG frames but the watermark can be extracted without the need of the original MPEG file. The scheme is based on MPEG audio layer 2.

The watermarking scheme makes use of three patterns to embed the watermark. One pattern encodes "0", one pattern encodes "1" and the third one, "sync-bit", is used for self-clocking and robustness against cropping (see Figure 2-6). The patterns are inserted into the scale factors by modifying the scale factors that differ from the pattern being inserted. The patterns are selected in a way that make the least amount of modifications in the scale factors. In a given region, the three patterns are searched for and counted. The one with the most hits is the embedded bit. For robustness against trimming, the sync-bits can be used. With fixed number of frames and a fixed number of bits to encode a letter, the sync bits can be used as a header to resynchronize the algorithm at the beginning of each watermark. From the test results, it has been shown that it is very hard to detect changes made by the algorithm even at higher bit rates than 14bps. But it was mentioned that the algorithm is not robust against decoding to PCM-Wave and back to MPEG.

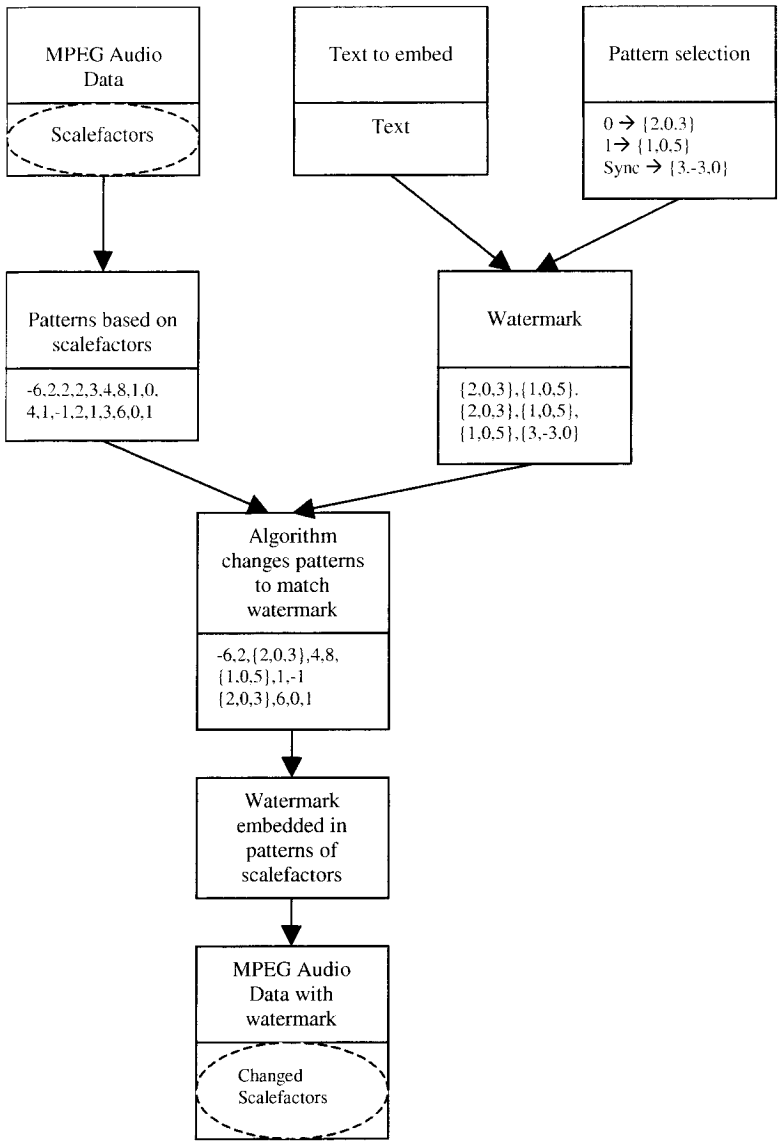


Figure 2-6 [30] watermarking scheme

Koukopoulos and Stamatiou [59] extended the work done in [30] by making the algorithm embed the watermark in the MPEG audio layer3. The algorithm made use of crypto-keys, using the 3-coloring problem [19], which are claimed to be robust against malicious attacks because they can be detected correctly, even if a big percentage of them has been damaged.

2.1.5. *3D Virtual Objects Watermarking*

The most important component for data embedding in both VRML and MPEG-4 is the 3D polygonal mesh. The shape of a 3D polygonal mesh is defined by two components, vertex coordinates and vertex topology. These components define more complex geometrical primitives that are, lines, polygons, and polyhedrons. These geometrical primitives have their own quantities such as length of a line segment and volume of a polyhedron. These are called geometrical quantities. These geometrical primitives also have their own topology, which is, for example, the connectivity of vertices and triangles [127]. These topology and geometrical quantities are the most important targets for watermark embedding in 3D polygonal meshes. These geometrical primitives typically have an additional non-geometrical quantity associated with them. Per-vertex color, per-face normal vector, per-vertex texture coordinate, per-face transparency, or per-volume refractive index, are examples of these non-shape-defining attributes. While less crucial than the former two shape-defining attributes, these non-geometrical quantities could be a good target for watermark embedding.

The following discussion will reveal some of the most important/recent algorithms developed so far for data embedding in 3D objects. These algorithms are based on the three different ways of data embedding in 3D objects: Geometrical, Topological and Non-Geometrical modifications.

Ohbuchi et al. [83] did a lot of research work on 3D object watermarking. One of the proposed schemes is called Triangle Similarity Quadruple (TSQ). The watermark is embedded by modifying the vertices of the 3D mesh. The algorithm makes use of four adjacent triangles which form a set called Macro-Embedding-Primitive (MEP). The MEP consists of a: Marker, Subscript and two Data variables to embed a watermark. The marker is used to detect that a watermark is embedded in that area, the subscript used to identify what watermarked data should be read first, and the data variables are used to store that watermark. The mesh vertices are modified according to whether the triangle

should contain a marker, a subscript or a data symbol (Figure 2-7). A watermarked mesh would contain multiple MEPs to embed a significant amount of data as shown in the example of Figure 2-8. Watermarks produced by the TSQ algorithm were claimed to withstand translation, rotation, and uniform-scaling transformations of the watermarked polygonal-meshes. The embedded message is resistant to resection and local deformation if it is repeatedly embedded over a mesh. The watermarks are destroyed, among other disturbances, by a randomization of coordinates, by a more general class of geometrical transformation, or by a topological modification such as re-meshing.

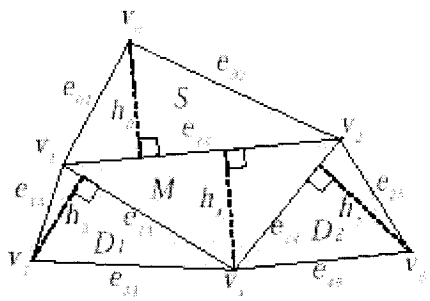


Figure 2-7 A macro embedding primitive[83]

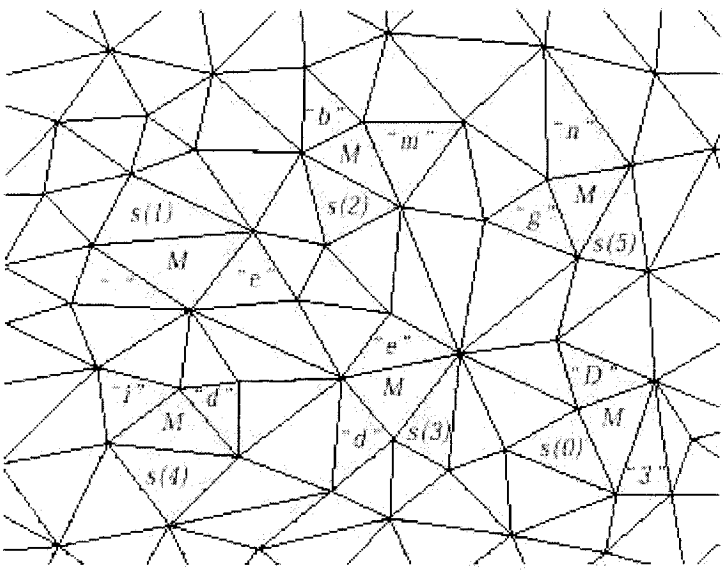


Figure 2-8 TQS watermarking example using six MEPs [83]

Another algorithm proposed by Ohbuchi et al. [84] makes use of the ratio of volumes of a pair of tetrahedrons as the embedding primitive. The algorithm is called Tetrahedral Volume Ratio (TVR) embedding. The watermark is embedded as follows:

A 3D mesh is first split into a sequence of triangles forming a triangle-spanning tree (Figure 2-9e). The centroid of the first triangle is used as a common apex (Figure 2-9f). This triangle, where the common apex is chosen from, is put aside so that its vertices are not modified later due to the embedding of the data. This common apex is used in forming tetrahedrons for the rest of the triangles in the triangle-spanning tree. A sequence of ratios of tetrahedral volumes is then generated (Figure 2-9g). A tetrahedron is used as the common denominator while the rest of the tetrahedrons are used as numerators. The ratio of the tetrahedral volumes is modified according to the symbol that needs to be embedded. For tests being done, it has been shown that the embedding data density ranged from 0.05 byte/ triangle to 0.2 byte/triangle. It has been claimed that the watermarks withstand affine transformation of the polygonal mesh. But the watermarks are destroyed by topological modifications such as resection/re-meshing, randomization of vertex coordinates, and geometrical transformations.

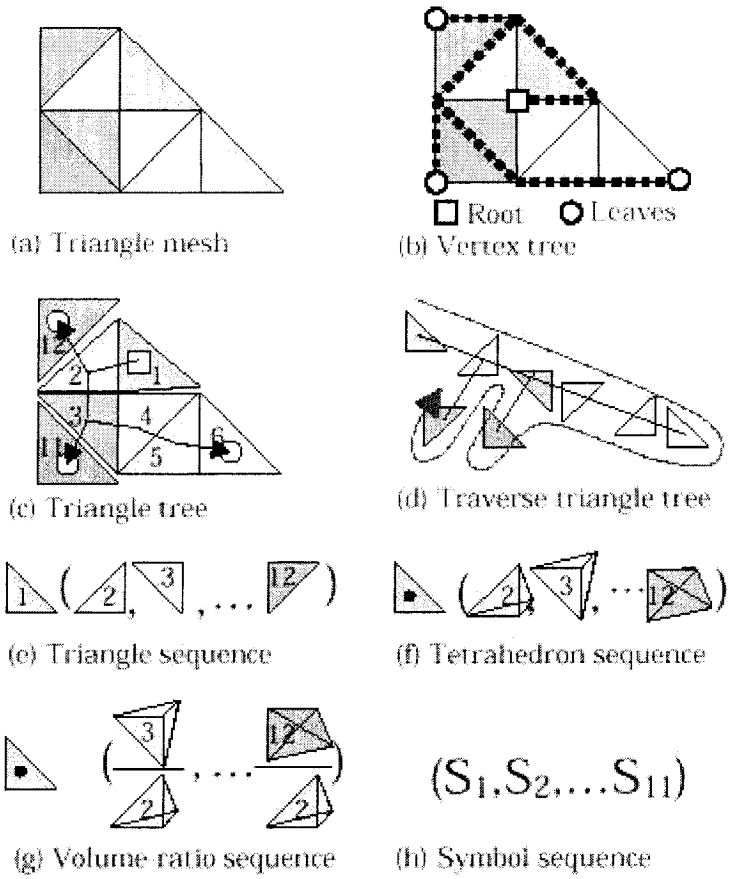


Figure 2-9 Embedding primitives using TVR embedding algorithm [84]

A third algorithm introduced by Ohbuchi et al. [84] is a public watermarking algorithm called Triangle Strip Peeling Symbol sequence (TSPS), which is based on a topological embedding primitive. To embed a binary bit string, starting from an edge e selected from the input mesh, a triangle strip S is developed and the direction of its growth (clockwise or counterclockwise) is oriented based on the message bit string. The triangle strip S is peeled off from M by splitting all the edges and vertices on the boundary of S except the initial edge e. The strip S is connected to the rest of the mesh only by the edge e, see Figure 2-10. Since the peeled strip caps the hole completely, proper colors and vertex normal vectors make the watermark invisible. Since both embedding primitive and arrangement are topological, watermarks produced by the algorithm are immune to geometrical transformation. Repetitive embedding makes the

watermarks resistant to resection. For tests, it has been found that the watermarks can be destroyed by topological manipulations, for example, by polygon simplification algorithms. A disadvantage of this algorithm is its low space efficiency compared to many algorithms based on geometrical primitives.

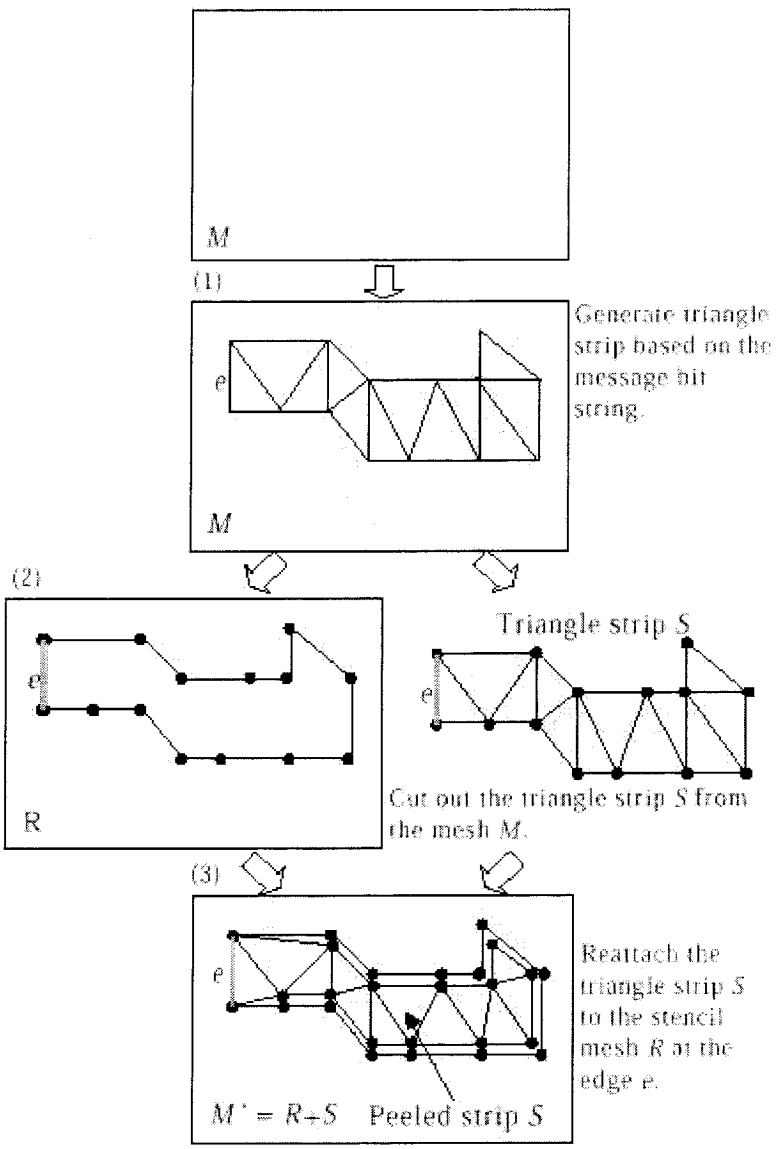


Figure 2-10 Triangle strip peeling symbol sequence algorithm [84]

Praun et al. [96] proposed a novel technique based on the principle of spread spectrum for watermarking. The approach is to convert the original triangle mesh into a multiresolution format [42][88], consisting of a coarse base mesh and a sequence of refinement operations, as shown in Figure 2-11. The m refinement operations that cause the greatest geometric change are selected. For each of these m refinements, a scalar basis function is defined over its corresponding neighborhood in the original mesh. The m basis functions form $\Phi = (\Phi^1,\ldots,\Phi^m)$ and are used to insert the watermark. Now for each watermarking coefficient w_i, a scalar basis function Φ^i_j is constructed over the mesh vertices v_j and associated with it a global displacement direction d_i. During the watermarking process, each watermarking coefficient w_i perturbs each vertex v_j by a vector proportional to $w_i \Phi^i_j d_i$. The watermark embedding process is expressed as a matrix multiplication: For each of the three spatial coordinates X, Y, and Z:

$$[v'_x] = [v_x] + \varepsilon * [\phi] * \begin{bmatrix} h_1 d_{1x} & \ldots & 0 \\ & \ldots & \\ 0 & \ldots & h_m d_{mx} \end{bmatrix} * [w]$$

where v'_x are the X coordinates of the watermarked mesh vertices

v_x are the X coordinates of the original vertices **v,**

ε is a user-provided global parameter that scales the energy of the watermark,

Φ is an nxm matrix with the scalar functions Φ^i as columns, hd_x is an mxm diagonal matrix whose entries are the X components of the displacement directions d_i scaled by the basis function heights h_i, and

w is the watermark.

By concatenating the rows of the matrices and vectors corresponding to the three coordinate components (X, Y, Z) the insertion process can be expressed as a single equation

v' = v + B * w

The original document **v,** along with the watermark w are stored and kept secret, and the watermarked document **v'** is published.

For the extraction and verification of the watermark, the original document is used and processed as follows:

$Bw^* = (v^* - v)$

where w^* is the extracted watermark,

$\mathbf{v}^*$ are the vertex coordinates of the resampled attacked mesh,

v are the vertex coordinates of the original mesh.

The inserted and extracted watermarks are then compared using statistical analysis for verification. The watermark has been claimed to be robust against reordering, addition of noise, cropping, smoothing, simplification and insertion of a second watermark.

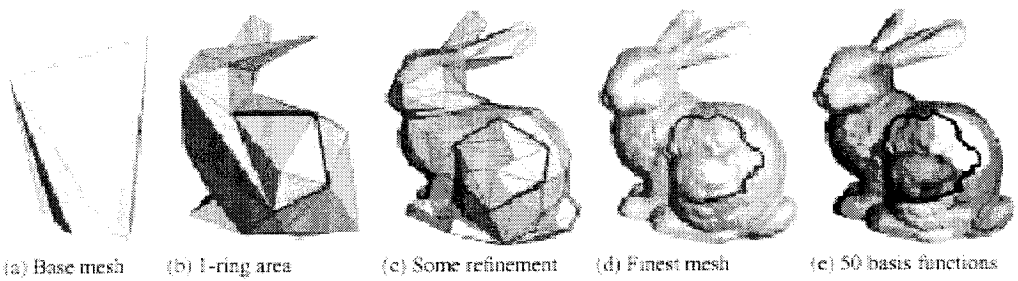


Figure 2-11 Progressive mesh representation [96]

Wagner [120] introduced a different algorithm for watermarking polygonal meshes. A mesh M={P,C} is given consisting of a set $P=\{p_i\}_{i=0...n}$ of n+1 points p_i with coordinates $\{x_i, y_i, z_i\}$ and connectivity information $C=\{\{i_k, j_k\}\}_{k=0,...,m}$ for $0 \le i_k \le n$, $0 \le j_k \le n$. C is a set of index pairs $\{i_k, j_k\}$ each of which corresponds to one edge of the mesh. Moreover, the star S_i of a point p_i can be defined as the set of indices of all points connected to p_i.

Now averaging the vectors from point p_i to all points connected by an edge is as follows:

$$n_i = \frac{1}{|S_i|} \sum_{j \in S_i} (p_j - p_i)$$

The average discrete normal vector length d can then be calculated.

$$d = \frac{1}{n+1} \sum_{i=0}^{n} \|n_i\|$$

where $\|n_i\|$ denotes the Euclidean norm of n_i.

Each individual normal vector length is converted into an integer t_i according to:

$$t_i = round\left(\frac{c}{d}\|n_i\|\right)$$

where c is an arbitrary but fixed real number.

The integers t_i remain unchanged if the mesh is subject to translation, rotation or scaling. Therefore, these numbers are used as the target for the data hiding procedure.

The watermark itself is a function $f(v)$ defined over a sphere. The function value of f at n_i is converted to an integer

$$w_i = round\left(2^b f\left(\frac{1}{\|n_i\|}n_i\right)\right)$$

The integer b denotes the number of bits used to store w_i.

For the embedding procedure, b bits are chosen from the binary representation of t_i and are replaced with the bits of the binary representation of w_i to obtain a modified value t_i' with embedded watermark. The robustness of the watermark strongly depends on the location of the inserted bits. In the final step of the encoding procedure, the point coordinates of the mesh are changed such that the new discrete normal vectors n_i' exhibit the watermark in their normalized length.

$$n_i' = \frac{t_i d}{c}\frac{n_i}{\|n_i\|},$$

and therefore the new point coordinates p_i.

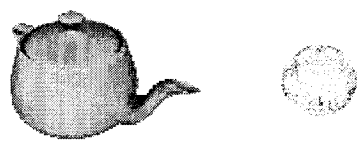
$$n_i' = \frac{1}{|S_i|}\sum_{j \in S_i}(p'_j - p'_i)$$

Decoding on the other hand, can be achieved in real time by calculating the values t_i and extracting the appropriate bits in which the watermark is hidden. Notice that since the normal vectors n_i have been changed, proper decoding requires that the main parameter c is adjusted to

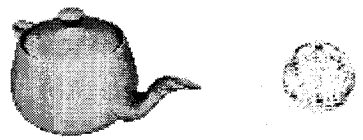
$$c' = c\frac{d}{d_{new}}$$

where d_{new} is the average normal vector length of the modified mesh.

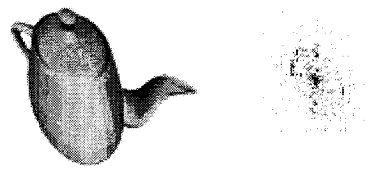
It has been shown that the watermark is robust against translation, rotation and scaling but not to affine transforms.



(a) mesh with the encoded watermark



(b) watermark after reducing the coordinates digit precision



(c) watermark after applying the affine transform

(d) watermark after deleting half of the mesh

Figure 2-12 Mesh and its corresponding watermark [120]

A new robust watermarking technique developed by Ohbuchi et al. [85] embeds the watermark in the 3D mesh spectral domain. The algorithm computes spectra of the mesh by using eigenvalue decomposition of a Laplacian/Kirchhoff matrix derived only from connectivity of the mesh. The Kirchhoff matrix K can be defined as follows:

$$K = D - A$$

where D is a diagonal matrix whose diagonal element $D_{ii} = d_i$ is a degree (or valence) of the vertex i, and A is an adjacency matrix of the polygonal mesh whose elements a_{ij} are defined as:

$$a_{ij} = \begin{cases} 1, & \textit{if vertices } i \textit{ and } j \textit{ are adjacent} \\ 0, & \textit{otherwise} \end{cases}$$

A mesh with n vertices produces a Kirchhoff matrix K of size nxn, whose eigenvalue decomposition produces n eigenvalues λ and n n-dimensional eigenvectors w_i ($1 \leq i \leq n$). The n mesh spectral coefficient vectors $r_i = (r_{s,i}, r_{t,i}, r_{u,i})$ are generated when projecting each of the vertex coordinate $v_i = (x_i, y_i, z_i)$ onto the i-th normalized eigenvectors e_i

$$e_i = w_i \,/\, \|w_i\|$$

The watermark is inserted using the idea of spread spectrum similar to that used by Cox et al. [21].

The algorithm has been claimed to be robust against similarity transforms (rotation, translation, and uniform scaling), mesh smoothing ("low-pass filtering" of 3D shapes), random noise added to vertex coordinates and resection of part of the mesh.

In [36], Hartung et al. proposed a spread spectrum method for watermarking of MPEG-4 Facial Animation Parameters (FAP) of synthetic faces. The watermarks are additively embedded into the animation parameters - the watermark is not contained in the waveform representation of a depicted object (the pixels) but in the semantics (the way the head and face move). Smoothing of the spread-spectrum watermark by low-pass filtering and adaptive amplitude attenuation prevents visible distortions of the animated head models. The watermarks cannot only be retrieved by correlation from the watermarked parameters, but also from video sequences showing 3D head models animated with the watermarked parameters, even after modifications such as block-based compression.

2.1.6. *Software Watermarking*

Very little work seems to be publicly available on watermarking software, apart from few US patents such as [25][41][80][103].

In [20], Collberg and Thomborson explain two ways of watermarking software: Static Software Watermarking and Dynamic Software Watermarking.

In static software watermarking two types of watermarks can be embedded: Data Watermarks and Code Watermarks. Static data watermarks can be embedded in the header sections, string sections, debugging sections, etc. Static code watermarks are stored by modifying the code according to the watermarking bit, which needs to be embedded. For example, having two statements S_1 and S_2, a watermarking bit can be embedded by flipping S_1 with S_2 without affecting the program functionality. The problem with static watermarks though is that they can easily be destroyed by obfuscating transformations, which scramble all the static structures of a program.

In dynamic software watermarking, three types of software techniques are defined [20]: Easter Egg Watermark, Data Structure Watermark, and Execution Trace Watermark. An Easter Egg watermark is a piece of code that gets activated for a highly unusual input to the application. This watermark technique is very interesting and has been used, but it can be located easily in the code and destroyed. Data structure watermark is embedded within the state (global, heap, stack data, etc...) of a program P as it is being run with a particular input I. The watermark can be extracted by examining the data after the program has finished executing using input I. But it has been claimed that this way of watermarking is susceptible to attacks by obfuscation, where one variable can be split into two variables and several variables can be merged into one. Work on improving this technique has been proposed by Collberg and Thomborson [20] by embedding the watermark in the topology of a dynamically built graph structure. The code that manipulates dynamic graph structure is hard to analyze because of the pointer aliasing effects, and hence semantics preserving transformations that make fundamental changes to a graph will be hard to construct. This dynamic graph topology watermarking scheme

has been developed by Palsberg et al. [86], called JavaWiz. It is written in Java and can watermark Java1.2 source programs. The watermarking software was tested against obfuscation and packaging attacks using the Java bytecode obfuscator WingGuard [123] and the Java packaging tool JAX [113]. It has been claimed that the watermark remained intact after the attacks. In execution trace watermarking, the watermark is embedded in the execution trace when the program is run with a defined input I. It has been shown that this watermarking technique can be destroyed by obfuscating an instruction trace. No published work has been done on watermarking using the execution trace scheme.

2.2. Benchmarking tools

2.2.1. Stirmark 3.1

Stirmark was the first benchmarking tool developed at the University of Cambridge for digital watermarking technologies. Given a watermarked input image, Stirmark generates a number of image modifications which can then be used to verify if the embedded watermark can still be detected or not. Image alterations implemented in Stirmark 3.1 include: cropping, flip, rotation, rotation-scale, FMLR, sharpening, gaussian filtering, random bending, linear transformations, aspect ratio, scale changes, line removal, color reduction, and JPEG compression.

2.2.2. Checkmark 1.0

Checkmark is a benchmarking suite for digital watermarking technologies developed at the University of Geneva. Running on Matlab under UNIX and Windows, it provides efficient and effective tools to evaluate and rate watermarking technologies. Checkmark contains some attacks not present in Stirmark. Moreover, it takes the watermark application into account which means that the scores from individual attacks are weighed according to their importance for a given watermark usage. Checkmark includes additional new classes of tests to those made by Stirmark 3.1 including: wavelet compression (JPEG 2000 based on Jasper), projective transformations, modeling of video

distortions based on projective transformations, warping, copy, template removal, denoising, denoising followed by perceptual remodulation, non-linear line removal and collage.

2.2.3. *Optimark*

Optimark is a new benchmarking tool [109] for still image watermarking that has just come out. It has been developed at the University of Thessaloniki, Greece. It can run on any machine with Windows 95/98/2000/NT/Me/XP. The types of attacks that can be generated by Optimark are: no attack, cropping, line and column removal, general linear transformation, scaling, shearing, horizontal flip, rotation, rotation + autocropping, rotation + autocropping + autoscaling, sharpening, gaussian filtering, median, and JPEG compression.

2.2.4. *Certimark*

Certimark (CERTIfication and WaterMARKing techniques) is a new benchmarking tool developed in 2002. The focus of the design and development of Certimark is on creating a complete benchmarking suite for still image and video watermarking technologies. The aim is at making this benchmark suite a reference for both technology suppliers and customers.

2.3. Digital Rights Management Standards

2.3.1. *STEP 2000/2001*

STEP 2000/2001 is an international evaluation project that promotes and recognizes those companies that have qualified successful implementation of digital music distribution based on digital watermarking technology.

MarkAny has proved and recognized its excellence of watermarking technology and was certified by STEP 2000 and STEP 2001.

Currently, STEP is composed of JASRAC, CISAC and BIEM. Among these organizations, the key project leader is JASRAC (Japanese Society for Rights of Authors, Composers and Publishers). JASARC coordinates, evaluates and announces the companies that have passed series of strict test, also, Nomura Research Institute (NRI) and Acoustic Laboratory are co-managers of the project test and evaluation for digital watermark technology.

2.3.2. SMDI

Secure Digital Music Initiative (SDMI) is an Internet record industry consortium for promoting the development of digital music file formats. It was founded to create an open technology specification to promote the sale of digital-form music on the Internet by worldwide major record labels as well as related organizations and corporations. It aims to develop technologies to protect music copyrights and prevent illegal copy.

SDMI is comprised of around 200 firms including the worldwide record labels, music providers, and copyright technology firms, which are stimulated by MP3 and free music service providers such as Napster and MP3.com. As an effort to limit peer-to-peer information sharing service, SDMI tries to select the world standard technology to embed watermarks to the music files for copyright protection.

So far, SDMI planed to get proposals of digital watermarking technology and select the best technology that satisfied with SDMI requirements. There are 12 companies that submitted the proposals, but only 4 companies remain.

2.3.3. MPEG

Moving Picture Experts Group (MPEG) founded by ISO/IEC is rolling out its standardization efforts through MPEG1, MPEG2, MPEG4, MPEG7 and MPEG21. The standardization activities have been almost complete for MPEG-4.

MPEG-4 Intellectual Property Management and Protection (IPMP) is a framework that allows the design of domain-specific (non-standardized) IPMP systems (IPMP-S). While MPEG-4 does not standardize IPMP systems, it does standardize the MPEG-4 IPMP interface. A more detailed explanation of MPEG-4 IPMP can be found in the appendix.

2.3.4. *cIDf*

cIDf was formed in August 2000 by Nippon Telegraph and Telephone Corp (NTT), Kyoto Digital Archives Organization, Dentsu Inc., Hitachi, Ltd., Matsusita Electric Industrial Co., Ltd., Sharp Corp., and PDC. Its main objective is to establish a framework for promoting digital content commerce, and the primary step to achieving this goal is to standardize the content ID, which is a unique code embedded in each piece of digital content for the purpose of copyright verification.

2.3.5. *OeBF*

Open Book Forum (OeBF) is international standardization organization in order to establish the standard of electronic books. With Microsoft working in a central, Intertrust, Adobe, Overdrive and others have been participated. OeBF announced OPBPS(Open eBook Publication Structure) 1.0 in 1999 and confirmed the standard of version 1.0.1 in February 2001.

OeBF, as AAP (Association of American Publisher) and EBX (Electronic Book Exchange) who are eBook standardization organization, announced eBook standard based in XML. OeBF is composed of 11 work groups. The IP Policy Committee is one of those groups that is taking part in making intellectual property policy, definition, and extracting debates in OeBF.

2.3.6. *CPTWG*

Copy Protection Technical Working Group (CPTWG) is the working group of copyright protection technology under DVD forum to set up the standard. CPTWG made up the working sub-group, Data Hiding Sub Group (DHSG). The DHSG is an ad hoc group of experts from computer, consumer-electronics and movie-studio industries that was formed to assess the technical merits of competing copy-protection proposals for movies, video and other content on digital media. The new framework, which controls recording and playback by means of watermarks, requires the standardization of watermarking for effective implementation in consumer DVD devices.

2.3.7. DOI

Digital Object Identifier (DOI) was started by AAP (Association of American Publisher). In July 1999, the National Information Standards Organization (NISO) passed the DOI standard and confirmed it as an official NISO standard, Z39.84-2000. DOI is used for identifying and exchanging intellectual property in the digital environment. It provides a framework for managing intellectual content, for linking customers with content suppliers, for facilitating electronic commerce, and enabling automated copyright management for all types of media. Using DOIs makes managing intellectual property in a networked environment much easier and more convenient, and allows the construction of automated services and transactions for e-commerce.

2.4. Steganographic/watermarking software developed

The following are some software tools developed for information hiding [87]:

DiSi-Steganograph: is a very small, DOS-based steganographic program that embeds data in PCX images.

EZStego: is a Java based steganographic software, which modifies the LSBs of still pictures (supports only GIF and PICT formats) and rearranges the color palette.

Gif-It-Up v1.0: is a stego program for Windows 95 that hides data in GIF files. It replaces colour indexes of the GIF colour table with indexes of 'colour friends' (a colour friend is a colour in the same table and as close as possible).

Hide and Seek: is a stego program that hides any data into GIF images. It flips the LSBs of pseudo-randomly chosen pixels. The data are first encrypted using the blowfish algorithm.

JPEG-JSTEG: hides data inside a JPEG file.

MandelSteg and GIFExtract: hides data in fractal GIF images. MandelSteg will create a Mandelbrot image (though it could be modified to produce other fractals), storing data in the specified bit of the image pixels, after which GIFExtract can be used by the recipient to extract that bit-plane of the image.

MP3Stego: hides data in popular MP3 sound files.

Nicetext: transforms ciphertext into innocuous text, which can be transformed back into the original ciphertext. The expandable set of tools allows experimentation with custom dictionaries, automatic simulation of writing style, and the use of Context-Free-Grammars to control text generation.

Pretty Good Envelope: hides data in almost any file. In fact, it embeds a binary message in a larger binary file by appending the message to the covert file as well as a 4-byte pointer to the start of the message. To retrieve the message, the last 4 bytes of the file are read, the file pointer is set to that value, and the file read from that point.

OutGuess: is a steganographic tool for still images. It support the PNM and JPEG image formats. OutGuess preserves statistics based on frequency counts. As a result, no known statistical test is able to detect the presence of steganographic content.

Stealth: is a simple filter for PGP, which strips of all identifying header information. Only the encrypted data (which look like random noise) remain; thus it is suitable for steganographic use.

Snow: is used to conceal messages in ASCII text by appending white spaces to the end of lines.

Steganography Tools 4: encrypts the data with IDEA, MPJ2, DES, 3DES and NSEA in CBC, ECB, CFB, OFB and PCBC modes and hides them inside graphics (by modifying the LSBs of BMP files), digital audio (WAV files) or unused sectors of HD floppies. The embedded message is usually very small.

Steganos: is an easy to use wizard style program to hide and/or encrypt files. Steganos encrypts files and hides them within various different types of files. It also includes a text editor using the soft-tempest technology. Many other security features are included.

Steghide: features hiding data in BMP, WAV and AU files, blowfish encryption, MD5 hashing of passphrases to blowfish keys and pseudo-random distribution of hidden bits in the cover-data.

Stegodos: is a set of DOS programs that encodes messages into GIF or PCX images. It works only with 320x200x256 pictures. The data embedded by modifying the LSBs of the picture (is noticeable in most cases).

Stegonosaurus: is a Unix program that will convert any binary file into nonsense text, but it statistically resembles text in the language of the dictionary supplied.

StegonoWav: is a Java (JDK 1.0) program that hides information in 16-bit wav files using a spread spectrum technique.

wbStego: lets you hide data in bitmaps, text files and also HTML files. The data are encrypted before embedding. Two different user interfaces are proposed: 'the wizard' guides the user step by step and the 'pro' mode provides full control.

Chapter 3

MPEG-4 Player

This chapter explains a developed MPEG-4 player. It is a prototype tool for the composition, editing, and viewing of MPEG-4 scenes. It is based on the COSMOS framework developed by [24].

The composition of scenes is very important and should follow the MPEG-4 standard for them to be rendered and visualized by the MPEG-4 player.

This chapter will first give a brief introduction of COSMOS framework that the player is based on, then the system architecture and the developed Graphical User Interface will be explained next.

3.1. COSMOS Overview

The Collaborative System based on MPEG-4 Objects and Streams (COSMOS) is a framework developed for the collaboration of virtual reality scenes. It was developed using pure Java code. It contains a VRML parser, BIFS encoder and decoder, as well as a Java 3D renderer. A Java 3D scene can be displayed directly. A VRML file is parsed and converted to Java3D before being displayed. A MPEG-4 BIFS file is decoded then displayed. All files displayed can be saved as BIFS files, as shown in Figure 3-1.

COSMOS implements many of the nodes specified in MPEG-4 BIFS in order to allow the encoding and decoding of BIFS streams. The additional compression achieved through the quantization of field's values and the BIFS-Anim protocol were not implemented. BIFS commands were sent using the BIFS-Update protocol.

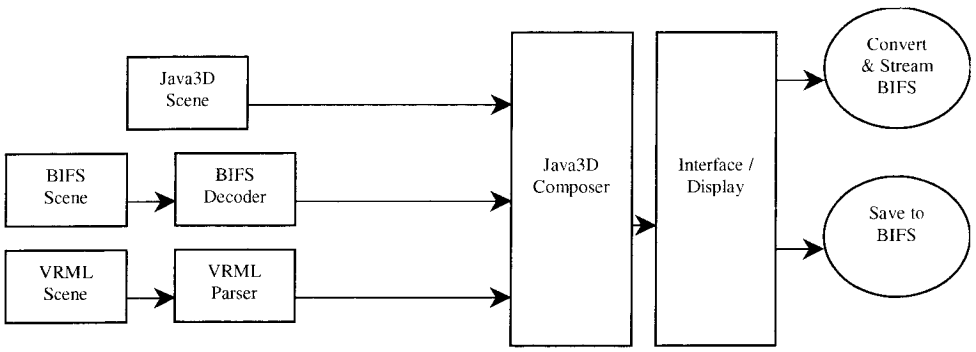


Figure 3-1. COSMOS framework

3.2. System Architecture Design

The COSMOS framework has been enhanced to support XMT-A scene formats. Unlike the MPEG-4 BIFS binary format, MPEG-4 XMT is an XML text-based format, which makes it easy to generate, edit and modify MPEG-4 scenes. Please refer to the appendix for a detailed explanation on the MPEG-4 standard.

The MPEG-4 player has been designed with added tools for the composition of MPEG-4 XMT scenes, an XMT parser, BIFS/XMT codec, a MPEG-4 XMT tree structure display, and the ability to manipulate a scene of a given object in the scene and update the modification in the MPEG-4 XMT tree structure display. Figure 3-2 shows the extra features added on top of the COSMOS framework and used in the MPEG-4 player. It should be noted that animations of objects or scenes are not supported in the player.

An MPEG-4 XMT scene can be generated from basic objects (i.e cube, sphere, cylinder, and cone) or importing one or more MPEG-4 XMT scenes and combining them together.

An MPEG-4 XMT scene can also be generated by saving a displayed VRML or MPEG-4 BIFS scene to the XMT format. It should also be noted that a VRML, or an XMT scene can also be saved in the MPEG-4 BIFS format.

Both the XMT parser and the XMT/BIFS codec were implemented using pure Java code. The XMT parser was built using the Document Object Model (DOM) of the Java API for XML Processing (JAXP). The generated DOM tree of a given parsed XMT scene is displayed (will be shown in the Graphical User Interface section next). Any manipulations applied on the displayed scene or a specific object in the scene will also be reflected in that tree.

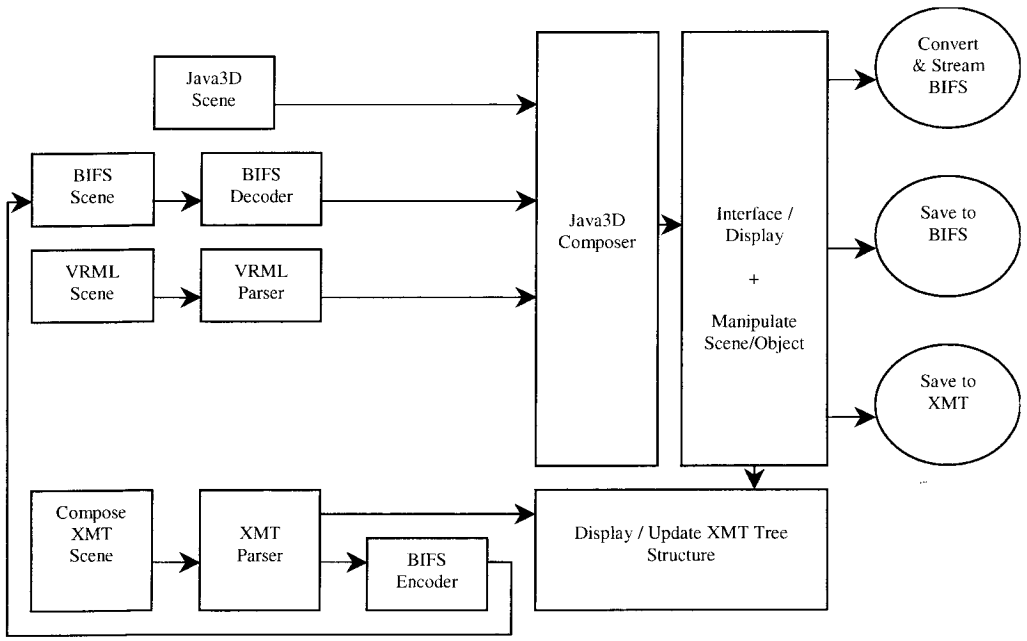


Figure 3-2. MPEG-4 Player Architecture

3.3. Graphical User Interface

Figure 3-3 shows the screen shot of the MPEG-4 player. All the features have been numbered and described as follows:

1- The Canvas3D where the 3D scenes are displayed.

2- The DOM tree of the displayed MPEG-4 XMT scene after being parsed.
3- Add a basic object (i.e. box, sphere, cylinder, or cone) to create a scene or add to it.
4- Import a VRML, BIFS, or a XMT scene to an existing scene. This feature will create a top grouping node to link the existing scene and the imported one to form one connected scene.
5- Open an existing scene (Java3D, VRML, BIFS or XMT format).
6- Save an opened/created scene.
7- Save as –filename of an opened/created scene.
8- View an existing file.
9- If selected, allows manipulation (translation, rotation, and/or scaling) of desired object or part of the scene in display. Any modifications will be updated automatically in the DOM tree shown in #2.
10- If selected, allows manipulation of the canvas itself for better view of objects or view hidden objects of a displayed scene.
11- Zoom in to the scene.
12- Zoom out of the scene.
13- Zoom the scene.
14- To move the whole scene to the left.
15- To move the whole scene to the right.
16- To move the whole scene upwards.
17- To move the whole scene downwards.
18- To center the whole scene in the middle of the canvas.
19- To delete all the create scene in display. This is also delete all the nodes and fields of that specific scene in the display DOM tree.
20- Will exit the application.

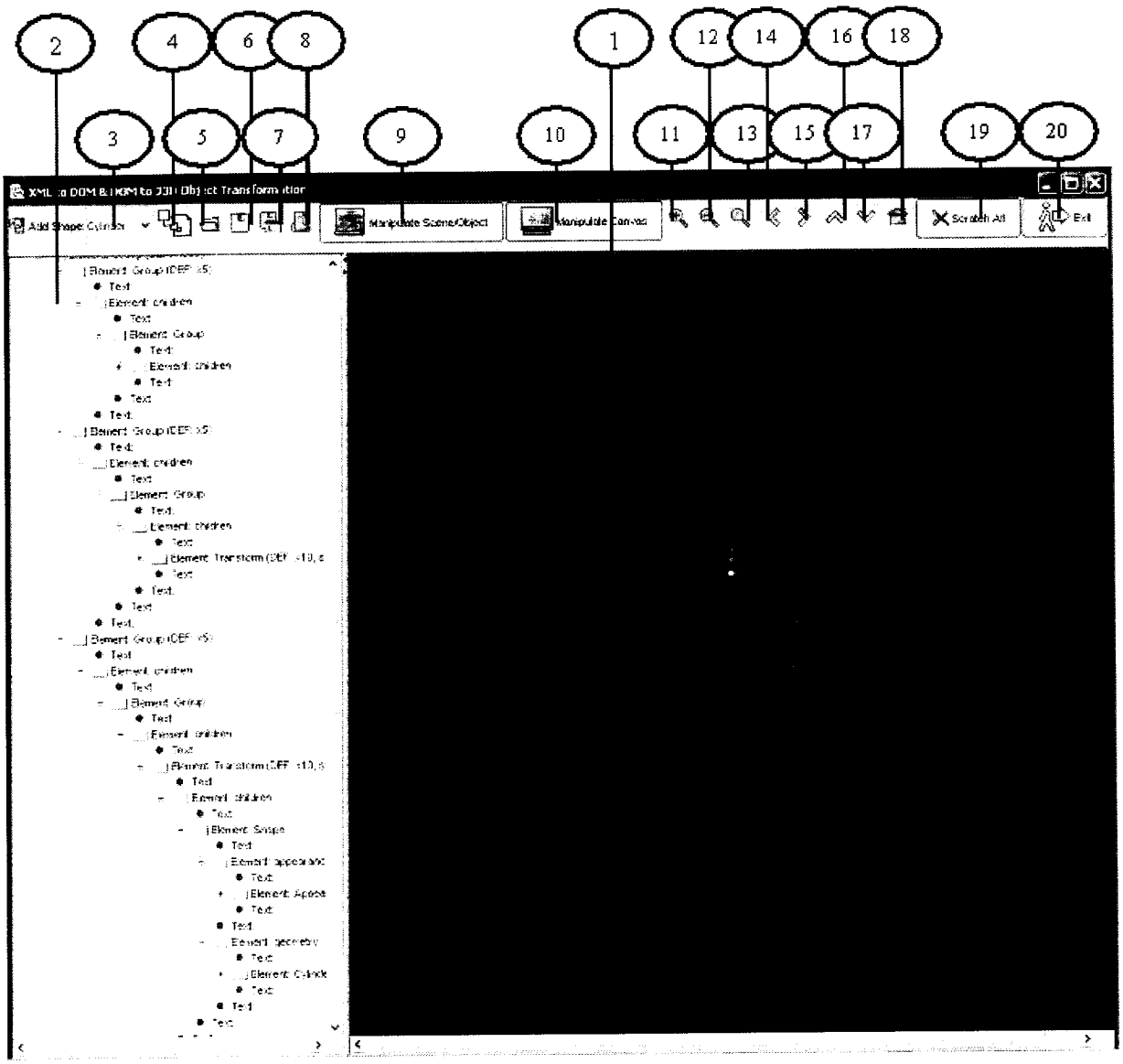


Figure 3-3. The GUI of the MPEG-4 player

Chapter 4

Robust MPEG-4 Content-Based Scene Structure Authentication and Content Protection

The previous chapter demonstrated a prototype software for the design and creation of MPEG-4 scenes. Those scenes can be in the MPEG-4 XMT textual format, or can be converted into the MPEG-4 BIFS compressed binary format for ease of distribution and/or streaming. But to protect those scenes from unauthorized use and distribution, a form of protection mechanism needs to be applied in order to authenticate and copyright those specific MPEG-4 scenes. This chapter explains in details a novel robust algorithm that, through a secret generated PRS code, is able to authenticate a given MPEG-4 scene, locate any watermarked media objects and embed/extract any watermark from any of those copyrighted media objects. The algorithm is applied to a MPEG-4 XMT scene structure and its media content. Even if the scene is in the MPEG-4 BIFS format, it can be converted to the MPEG-4 XMT format and then checked for authenticity and any located copyrighted material, without affecting the protection algorithm in detecting a given suspicious scene.

In most media watermarking schemes, the watermark is usually inserted into the noise components of the watermarked object. In the Extensible MPEG-4 Textual format (XMT), however, which is a textual representation using the eXtensible Markup Language (XML) for representing MPEG-4 Systems and media content, whole new challenges and associated tradeoffs need to be considered. One characterizing main difference is the lack of bandwidth, derived from the inherent lack of major noise components in that domain. Considering the XMT format, a new protection mechanism needs to be applied to the XMT scenes. Having the XMT scenes as structured text-based data with media content, the protection mechanism can be applied to the structured text, using a novel approach and/or a watermark can be embedded in the media content, such as images and video, using one of the traditional watermarking techniques as shown in Figure 4-1.

The algorithm makes use of a unique secret generated pseudo-random sequence (PRS) code, which is a common code used for the MPEG-4 XMT scene structure authentication, and media content location and protection. A PRS is used instead of any randomly generated code due to the unique sequence bits, which makes the algorithm more robust against structural modifications and allows for error detection and location as will be explained later in details. Using the PRS code, the algorithm generates two content-based signatures for a given MPEG-4 XMT-A scene. The first signature is used to detect and authenticate the scene structure. This generated signature is content fragile and any modification to the structure's content will be detected, yet the algorithm applied with that generated signature is robust against structure modifications and able to detect a scene even after major scene structure tampering/modification. The algorithm makes use of MPEG-4 BInary format for Scenes (BIFS) compression standard for the generation of such a scene signature. The second generated signature, after applying a similar technique, is used to locate specific fields that contain the media data that need to be protected. With the same generated PRS code used for the protection of a given MPEG-4 XMT-A scene, the media data, i.e. pictures, texture images, are watermarked. The

watermarking algorithm for the media data makes use of spread spectrum watermarking proposed by Cox et al. [21] and Hartung et al. [37][38], which will be explained in details in chapter 6. Even if only part of a scene with its content has been unlawfully used, using the secret generated PRS code, the algorithm can still detect the scene, locate and extract the watermark from the media content and link the MPEG-4 scene with its content to the rightful owner.

But before proceeding to the proposed MPEG-4 protection algorithm, a brief description of the related works in this field is given.

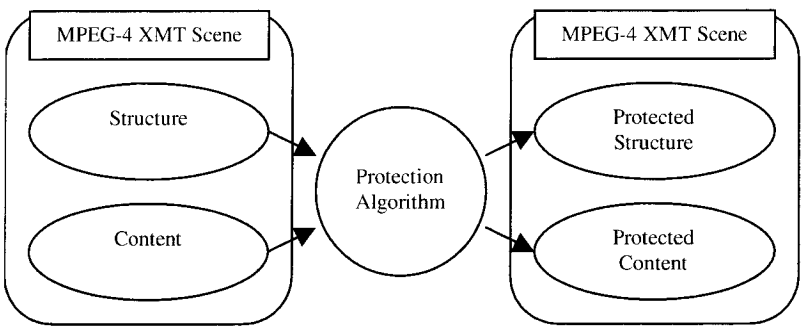


Figure 4-1 General MPEG-4 XMT scene protection

4.1. Related Work

As of the writing of this thesis, only the following published papers have to do with information hiding in XML structured text.

Inoue et al. [45][46] proposed several steganographic techniques for hiding information in XML data. One of the methods has to do with XML empty element tags. An empty element can be represented either by a start-tag immediately followed by an end-tag or an empty element tag [126]. By switching those two tag representations, a zero or a one bit can be embedded. Other methods in hiding information in XML data can be done using white spaces in tags, appearing order in elements, appearing order in attributes, and the order of elements containing other elements. All those methods proposed though are not

robust against watermarking attacks and the hidden information can simply be removed through node text formatting. These methods are suitable for steganographic purposes but not for watermarking means.

In[106], Sion et al. focus on a robust labeling algorithm for semi structured data. The watermark will still be detected if any minor modification to the data structure is made. The watermark is embedded in the media content (i.e. image, video, etc.) placed in the XML scene using a traditional content noise injection watermarking algorithm.

Lang et al. [63] proposed some general theoretical ideas for the protection of the MPEG-4 scene description and the content of the scene. No detailed algorithm or any experimental results were provided to support the feasibility/practicality behind those ideas.

No robust protection algorithm has yet been revealed that focuses on the protection of an MPEG-4 scene structure with its media content and be able to link their relationship through a common secret generated code, which is the focus of this research work.

4.2. MPEG-4 Proposed Protection Algorithm – Structure and Content

This section explains the proposed robust algorithm for authentication and copyright protection of MPEG-4 XMT-A scene structures and their media content. It should be noted that the proposed algorithm is designed to withstand major scene structure modifications and still be able to authenticate a given scene and locate the watermarked media content. Some of the modifications on the XMT structures would be: splitting the structure into multiple independent usable structured scenes, addition/elimination of insignificant nodes in the scene, and modification of node content within usability limits. In all modifications, the main idea is to try to destroy the structure properties while still preserving the usability of the given scene.

A general architecture of the proposed protection system is displayed in Figure 4-2 and Figure 4-3. With a privately (secret) generated PRS, a unique MPEG-4 scene content-based signature is obtained, shown in Figure 4-2, after passing a specific MPEG-4 XMT-A scene structure through the structure protection algorithm (explained below in details), which is kept secret for later detection/authentication of a suspicious MPEG-4 XMT-A scene structure. With the same private PRS, the media content (images, pictures, texture mapping, etc.) located within a MPEG-4 scene that needs to be protected, are passed through the watermarking algorithm to obtain the watermarked media objects. The media content signature is then generated after passing the scene structure through the content location algorithm. With the media content signature, the protection algorithm is able to locate the watermarked media objects in the scene even after major structure modifications.

The detection and authentication system for MPEG-4 scenes are revealed in Figure 4-3. Having a suspicious MPEG-4 scene, (if it is not in the XMT-A format, it should be converted before applying the algorithm), with the aid of the secret PRS and the scene structure signature, the structure protection algorithm is then able to detect and authenticate that given scene. If the scene was detected and related to the suspected one, the algorithm then generates a labeling sequence for that given scene. With that labeling sequence, when combined with the secret scene structure signature and the media content signature, the algorithm is able to locate all of the watermarked media content, even if the MPEG-4 scene structure has undergone structural modifications.

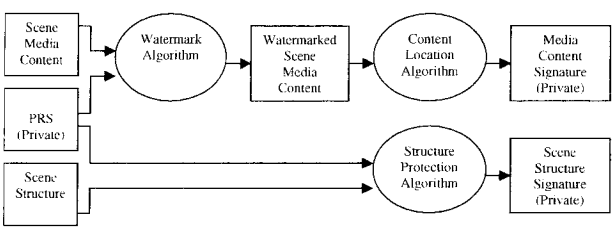


Figure 4-2. Protection of structure and content of an MPEG-4 XMT-A scene

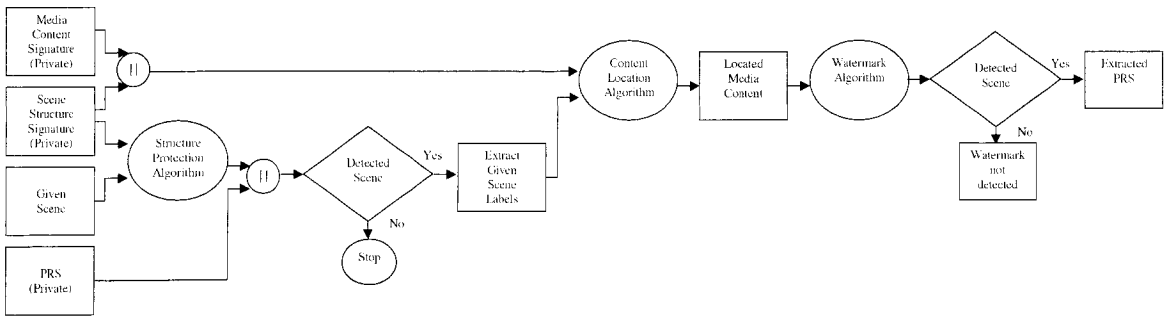


Figure 4-3. Authentication and copyright detection of an MPEG-4 scene

Having located the watermarked media objects, the watermarking algorithm is then able to detect the embedded PRS watermark and extract it, which is the same PRS used to generate the scene structure signature and media location signature. With this protection algorithm, it is easy to link the MPEG-4 scene structure and its media content to the same true owner.

4.2.1. MPEG-4 XMT-A Scene Structure Protection and Media Content Location Algorithms

This section provides a detailed explanation of the algorithms used for MPEG-4 scene structure protection and media content location.

4.2.1.1. Labeling

In order to identify and secure MPEG-4 XMT-A structured scenes, some form of node labeling should be considered. It can provide unique information in identifying and marking certain nodes that can still be detected even after structure modification attacks. However, considering the fact that the labeling method should withstand graph structure modifications due to tampering attacks and still be able to identify the labeled nodes,

would render most of the graph labeling algorithms [2][3][28][33][54][71] useless in that domain.

A solution that would make node labeling more robust to structural changes would be to consider the XMT scene structure as composed of nodes and their corresponding fields. The nodes are first labeled as follows:

$$\begin{aligned} L1(Node_{(0)}) &= seed + MPEG4Node_{(0)}Type \\ L1(Node_{(1)}) &= L1(Node_{(0)}) + MPEG4Node_{(1)}Type + \delta \\ L1(Node_{(n)}) &= L1(Node_{(n-1)}) + MPEG4Node_{(n)}Type + \delta \end{aligned} \quad (4\text{-}1)$$

where:

seed: a starting integer value.

n: node number in a MPEG-4 XMT tree.

Adding the MPEG-4 node type to the node's label, would make the label unique and adds more robustness to the scene under structural attacks.

The fields are then labeled with respect to their parent node as follows:

$$\begin{aligned} L1(Field_{(0)Node_{(n)}}) &= L1(Node_{(n)}) + \gamma \\ L1(Field_{(m)Node_{(n)}}) &= L1(Field_{(m-1)Node_{(n)}}) + \gamma \end{aligned} \quad (4\text{-}2)$$

where:

L1: general labeling algorithm used to label each node/field and identify it.

δ, γ: constant integer increment factors.

n: node number in a MPEG-4 XMT tree.

m: field number of a specific node.

As it can be seen, the fields' labels are directly related to their parent's node label and each other's labels and not to other node labels. This will make the algorithm more robust to structural modifications as it will be demonstrated in the next section.

A second labeling algorithm is then applied to a select number of fields. The labeling method is slightly different between fields containing bit information used for structure authentication and fields containing watermarked media content.

Field Labeling – Structure Protection

The fields that contain the required bit information are then selected. A second labeling algorithm is then used to label those selected fields. This labeling method is generated based on the field's label with respect to its surrounding nodes labels and the entire structure. The second labeling algorithm can be defined as follows:

$$L2_{protect}(Field_{(m)Node_{(n)}}) = \alpha * L1(Field_{(m)Node_{(n)}}) + \beta * \sum_{\#surround(Node_{(x)})} L1(Node_{(x)}) \qquad (4\text{-}3)$$

where:

$L2_{protect}$: a second labeling algorithm used on selected fields with required bits for structure authentication.

α: determines the weight of a certain field – value can range between 0.1 and 1.0.

β: provides more control and stability for a field with respect to its node neighbors – value can range between 0.1 and 1.0.

m,n: field with index m for a given node with index n.

x: index of a node surrounding a specific field.

Field Labeling – Content Location.

Fields located at the extremities of the XMT scene structure are not suitable for structure protection since they can easily be removed without affecting the rest of the structure. But most of the media carrying fields are at the extremities of the scene structure. This field

labeling algorithm will facilitate the detection of the selected fields containing the media data even under structure modification. For content protection, a robust watermark will need to be embedded in the media content, which can later be extracted in case of any suspicious and unauthorized use.

Since most of the media carrying fields are at the extremities of the structure, the second labeling algorithm makes use of the parent node and the node(s) above in the structure to generate the label for every selected field, as follows:

$$L2_{\text{content-locate}}(\text{Field}_{(m)Node_{(n)}}) = \alpha * L1(\text{Field}_{(m)Node_{(n)}}) + \beta * \sum_{x=n-1}^{n} L1(\text{Node}_{(x)}) \quad (4\text{-}4)$$

where:

$L2_{content\text{-}locate}$: a second labeling algorithm used on selected fields containing media data objects.

α: determines the weight of a certain field – value can range between 0.1 and 1.0.

β: provides more control and stability for a field with respect to its node neighbors – value can range between 0.1 and 1.0.

m,n: field with index m for a given node with index n.

x: index of a node surrounding a specific field.

The values for α and β can be adjusted according to the stability and robustness required. By increasing α for instance, greater labeling stability can be achieved. On the other hand, increasing the value of β makes a field's label more robust with respect to its neighboring nodes. Certain tradeoffs need to be considered on selecting those two values to achieve optimum results.

4.2.1.2. Signature Generation

This section deals with the details involved in generating the signatures used for MPEG-4 XMT-A scene structure authentication and media content location.

Scene Structure Authentication Signature

To be able to authenticate the scene structure and detect any modifications being applied to it, a scene authentication signature is generated. The algorithm makes use of Pseudo-Random Sequences (PRS) encoding and the MPEG-4 BInary Format for Scenes (BIFS) compression standard.

Given a primitive polynomial $h(x)$ of degree m with coefficients from a Galois Field with q elements GF(q) [7][10], a pseudo-random encoded sequence can be generated [4][18][23][31][40][99]. This pseudo-random sequence generated will be used to gather unique information from the XMT structured scene. For example, if $q = 2$, $m = 4$, $h(x) = x^4 + x^3 + 1$, the following PRS will be generated:

PRS: 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1

Having a window of size m slide across that sequence is unique [89]. By mapping this information to data gathered from the structured scene, then any modification to the structure will damage part of this continuous sequence and can therefore be detected and located. Moreover, even if an attack/modification got applied to the scene, the scene will still be detected and recognized.

To map the PRS to the scene structure, the algorithm makes use of the MPEG-4 BIFS compression standard. In the BIFS standard, the fields of each node are labeled uniquely using the minimum number of bits to achieve high compression ratio for BIFS scenes [24][43]. For our algorithm, the focus is only on fields with DEF IDs. For example, the Cone node shown in Figure 4-4 has four fields each with their corresponding DEF ID. To

identify and protect the scene, the algorithm goes through the XMT structured graph and selects those fields with DEF IDs that correspond to the PRS bits that were generated and that are not located at the extremities of the XMT-A structure. If a field cannot be found with a DEF ID that corresponds to the required bits, a fake node can be inserted into the structure that carries a field with that required DEF ID. All the nodes in the structure with their corresponding fields are then labeled using the $L1(Node_{(n)})$ and $L1(Field_{(m)Node_{(n)}})$, equations 4-1 and 4-2 given in section 4.2.1.1. The selected fields are then labeled again using the equation 4-3, $L2_{protect}(Field_{(m)Node_{(n)}})$, using an appropriate value for both α and β, where m is the index of a chosen field belonging to node n. For example, given part of a generated PRS "001000", information can be gathered from the XMT scene, shown in Figure 4-5, by selecting the fields {children, appearance, material} with DEF IDs {001, 0, 00} respectively. Not every node along the path needs to have a field that is labeled for structure identification. The labels generated $L2_{protect}(Field_{(m)Node_{(n)}})_{(i)}$ and the number of bits matches to some bits in the PRS in each field - $DEFNum(Field_{(m)Node_{(n)}})_{(i)}$, where i is the index of the saved fields Labels/DEFNums – are then saved. This will be the structured scene unique signature for later identification/authentication. The algorithm can be applied to different parts of the structure to increase the robustness of the structure against different kinds of attacks. The signature generated and the PRS sequence should remain secret and each can be encrypted with a symmetric secret key, as shown in Figure 4-6, for later detection/authentication of a given scene if required.

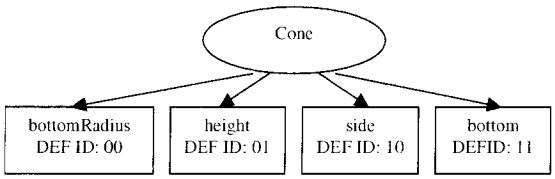


Figure 4-4. Node with its fields.

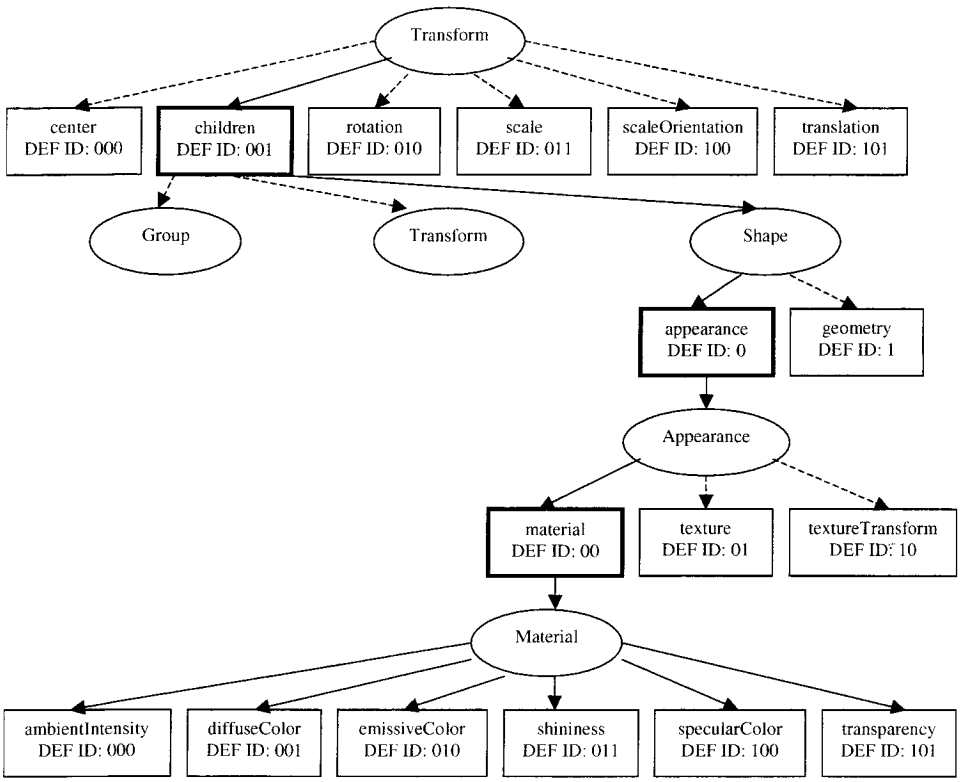


Figure 4-5. Sample of a XMT-A scene structure labeling.

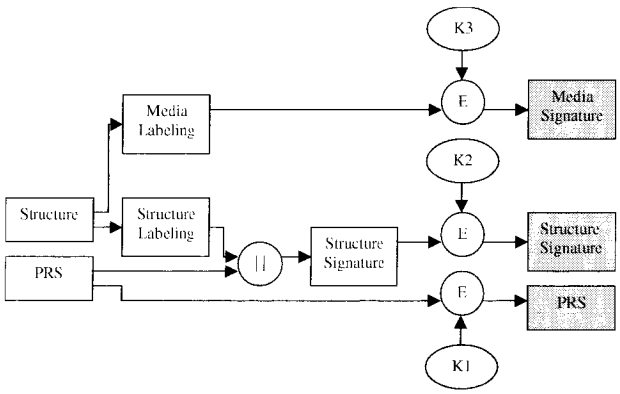


Figure 4-6. Generation of unique scene structure signature.

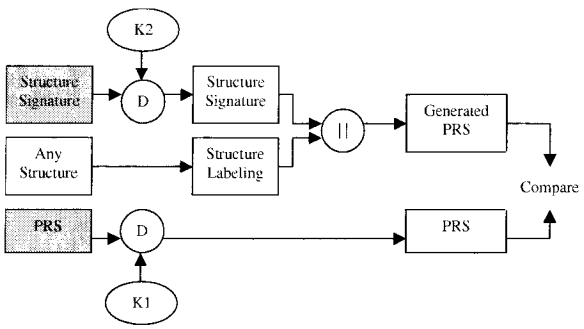


Figure 4-7. Detection and identification of a scene.

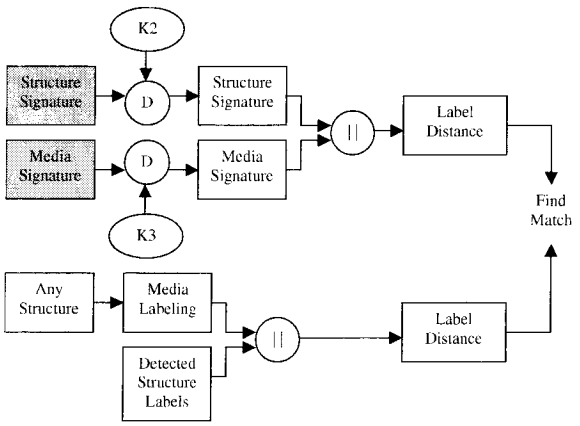


Figure 4-8. Detection of media carrying fields.

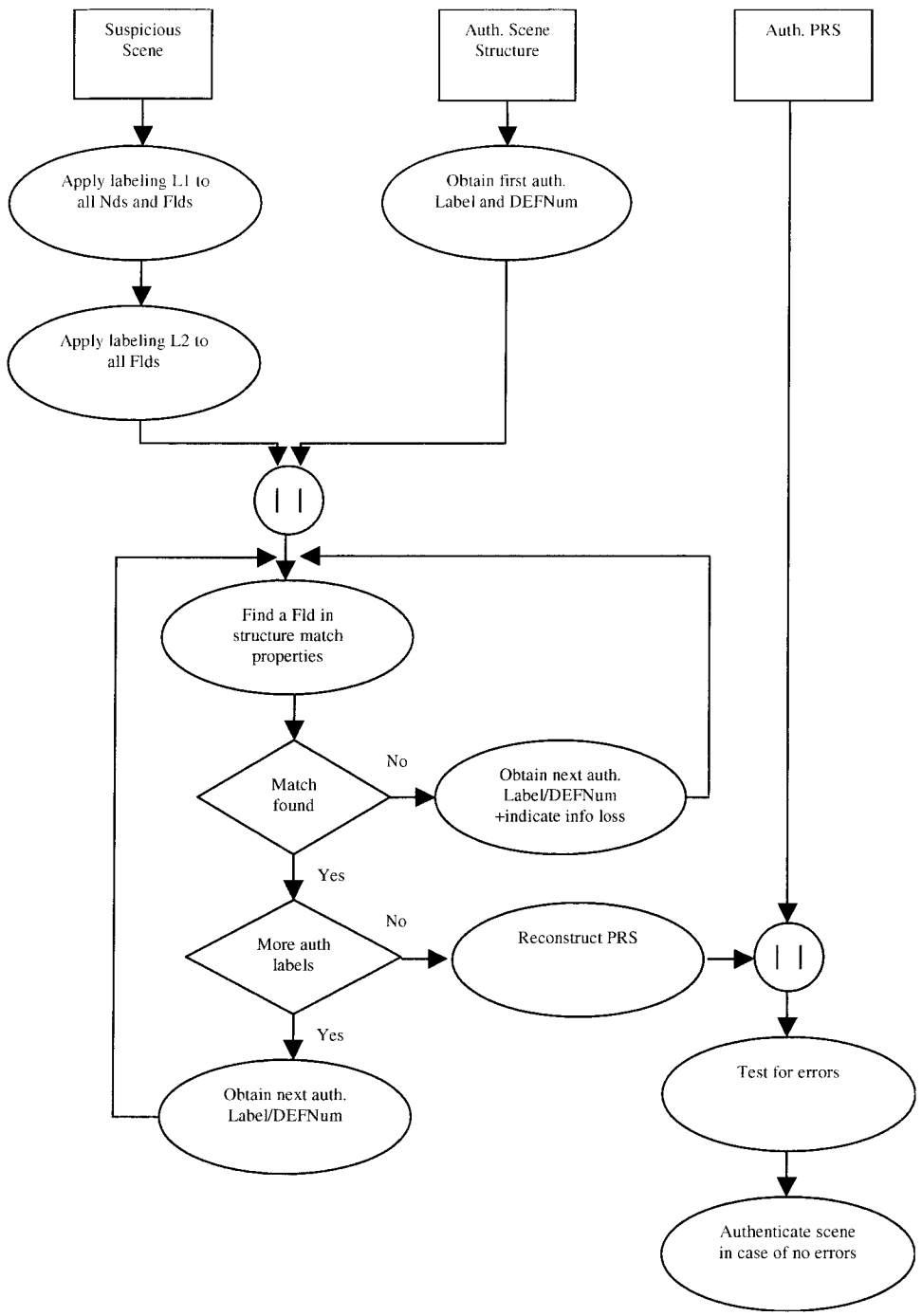


Figure 4-9. Detection and authentication algorithm for MPEG-4 scene structure.

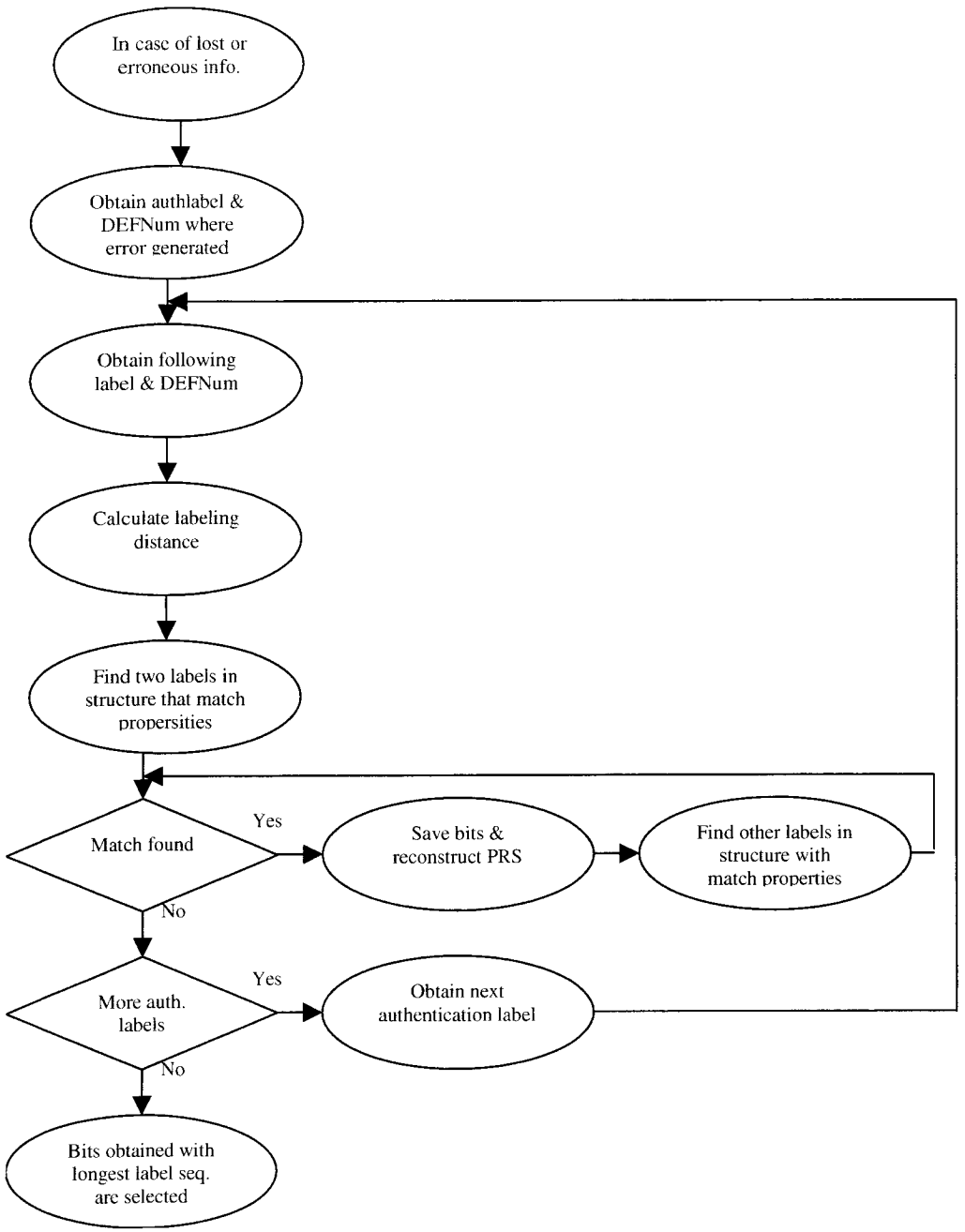


Figure 4-10. Detection/authentication algorithm for MPEG-4 scene in case of lost or erroneous data.

Content Location Signature

Since most of the media carrying fields are located at the extremities of the XMT-A structure, might contain no DEF IDs, and need to be specifically selected and not randomly based on the generated PRS as done for the structured authentication signature, the content location signature needs to be slightly modified from that of the authentication signature to meet the requirements. All the nodes in the structure with their corresponding fields are labeled using the $L1(Node_{(n)})$ and $L1(Field_{(m)Node_{(n)}})$ equations 4-1 and 4-2 given in section 4.2.1.1, as done for the authentication signature generation. The selected fields are then labeled again using equation 4-4, $L2_{content\text{-}locate}(Field_{(x)Node_{(y)}})$ for content location, using an appropriate value for both α and β, where x is the index of a chosen field belonging to node y. All the labels generated $L2_{content\text{-}locate}(Field_{(x)Node_{(y)}})_{(i)}$ will be the saved media signature (with each label at a specific index i) and encrypted if necessary, as shown in Figure 4-6, for later detection of those specific media carrying fields.

4.2.1.3. Scene Detection and Content Location

In the event of suspicious unauthorized use of an MPEG-4 scene, the scene is first converted to the XMT-A format and the following algorithm is then applied.

Scene Structure Detection/Authentication Algorithm

Given a suspicious scene, the structure signature is decrypted with the secret key and with the suspected scene structure a PRS is generated, as shown in Figure 4-7. With the second private key, the secret PRS is decrypted and compared to the generated PRS. The comparison will determine whether this suspected scene is the original copyrighted scene or not. General flowcharts of the detection and identification algorithm of a scene are shown in Figure 4-9 and Figure 4-10. A detailed explanation of the algorithm is as follows [104] (it should be noted that the word "embedding" is used for the fields selected and used in the signature generation process, and the words "extracting" or

"detecting" are used for fields selected and used from the suspected scene structure in the detection/identification process):

(1) Obtain the suspicious structured scene.

(2) Apply the labeling methods $L1(Node_{(n)})$ and $L1(Field_{(m)Node_{(n)}})$ (equations 4-1 and 4-2) to all nodes and fields in the XMT-A scene structure.

(3) Apply the labeling method $L2_{protect}(Field_{(m)Node_{(n)}})$ (equation 4-3) to "all" the fields of the scene structure.

(4) Compare all the labeled field values to their matching labeled field value selected during embedding (field selection process), $L2_{protect}[(Field_{(m)Node_{(n)}})]_{detecting} = L2_{protect}[(Field_{(x)Node_{(y)}})_{(i)}]_{embedding}$. It should be noted that m is a specific field of a given node n in the suspicious scene structure, and x is a specific field of a given node y found at location i of the structure signature.

(5) If there is a match between the embedded field label and the extracted field label, then check if the number of bits of the DEF ID in the extracted field matches with the number of bits in the field used for embedding $DEFNum[(Field_{(m)Node_{(n)}})]_{detecting} = DEFNum[(Field_{(x)Node_{(y)}})_{(i)}]_{embedding}$

(6) If there is a match, the DEF ID bits are assigned for reconstructing the m-sequence. If there is no match, an error is detected.

(7) The primitive polynomial $h(x)$ of degree m with coefficients from $GF(q)$, used to generate the PRS in the embedding process, is now used to check for the correctness of the extracted sequence and locate the erroneous data.

In the case of erroneous or missing bits from the extracted sequence (caused due to structure tampering/modification), an error correction/elimination process is done by applying an exhaustive search to the scene structure provided to locate the erroneous/missing data.

(8) Obtain the $L2_{protect}[(Field_{(x)Node_{(y)}})_{(i)}]_{embedding}$ and $DEFNum[Field_{(x)Node_{(y)}})_{(i)}]_{embedding}$ where the erroneous/missing bits were detected in the extracted sequence.

(9) Obtain the next label and bit number of the field used for PRS embedding, $L2_{protect}[Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding}$ and $DEFNum[Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding}$, where field x' of node y' indicate a different field located at index $i+1$ of the structure signature.

(10) Calculate the distance between the two labels; Distance = $L2_{protect}[(Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding} - L2_{protect}[(Field_{(x)Node_{(y)}})_{(i)}]_{embedding}$.

(11) Given the bit number of the two consecutive fields used for sequence embedding, $DEFNum[(Field_{(x)Node_{(y)}})_{(i)}]_{embedding}$ *and* $DEFNum[Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding}$ the expected/correct sequence bits can be obtained from the generating polynomial $h(x)$.

(12) An exhaustive search is made through the scene structure trying to find two fields with labeling distance that matches the distance of the labeled fields used for embedding; $L2_{protect}[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting} - L2_{protect}[(Field_{(m)Node_{(n)}})_{(k)}]_{detecting} = L2_{protect}[Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding} - L2_{protect}[Field_{(x)Node_{(y)}})_{(i)}]_{embedding}$, It should be noted that indices k and $k+1$ in the detecting process indicate the different structure location of two different fields m and m', whereas indices i and $i+1$ in the embedding process indicate the fields x and x' location in the structure signature.

(13) If there is a match in the labeling distance, check the DEF ID of those two detected fields with the expected bits obtained from the generating polynomial using the $DEFNum[(Field_{(x)Node_{(y)}})_{(i)}]_{embedding}$ *and* $DEFNum[Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding}$.

(14) If there is a match, save the bits obtained in a memory location $\text{Mem1(Rbits)}_{(a)}$ and save $L2_{protect}[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting}$ in a different memory location $\text{Mem2}(L2_{protect}[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting})_{(a)}$, where a is the index of the memory location.

(15) Another search is made through the structure to find another two fields with labeling distance that matches the distance of the labeled fields used for embedding

and DEF IDs that match those from the polynomial sequence (repeat from step12);

$L2_{protect}[(Field_{(m')Node_{(n')}})_{(p+1)}]_{detecting}$ - $L2_{protect}[(Field_{(m)Node_{(n)}})_{(p)}]_{detecting}$ = $L2_{protect}[Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding}$ - $L2_{protect}[Field_{(x)Node_{(y)}})_{(i)}]_{embedding}$

(16) If there is a match, save the bits obtained in the next position in the memory $Mem1(Rbits)_{(a+1)}$ and $L2_{protect}[(Field_{(m')Node_{(n')}})_{(p+1)}]_{detecting}$ in $Mem2(L2_{protect}[(Field_{(m')Node_{(n')}})_{(p+1)}]_{detecting})_{(a+1)}$

(17) The search continues until there are no more matches.

(18) The distance between the next two embedded field labels is considered; $L2_{protect}[Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding}$ and $L2_{protect}[Field_{(x)Node_{(y)}})_{(i+2)}]_{embedding}$, and the same process is repeated starting from step 8.

(19) If there is a match between $L2_{protect}[(Field_{(m)Node_{(n)}})_{(k+2)}]_{detecting}$ - $L2_{protect}[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting}$ = $L2_{protect}[Field_{(x)Node_{(y)}})_{(i+2)}]_{embedding}$ - $L2_{protect}[Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding}$, and the DEF IDs with the expected polynomial sequence bits, then search for the field label $L2_{protect}[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting}$ in the memory $Mem2(L2_{protect}[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting})$. If the label is found in the memory, i.e. $Mem2(L2_{protect}[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting})_{(a)}$, and is directly related with other detected labels, then replace the field label by the next field label $Mem2(L2_{protect}[(Field_{(m)Node_{(n)}})_{(k+2)}]_{detecting})_{(a)}$ and concatenate the bits obtained to the previous ones located in $Mem1(Rbits)_{(a)}$.

(20) The search continues for finding matching distances between field labels in the structured tree and the embedded field labels.

(21) When there are no more matches, the distance between the next two embedded fields is considered $L2_{protect}[Field_{(x)Node_{(y)}})_{(i+2)}]_{embedding}$ and $L2_{protect}[Field_{(x')Node_{(y')}})_{(i+3)}]_{embedding}$, and the process is repeated until there are no more consecutive embedded fields to consider.

(22) The concatenated bits found in Mem1(Rbits) with the longest corresponding labeling sequence located in Mem2, are considered the most authentic and matching to the true embedded sequence, since the consecutive labeling sequence of them matches those of the embedded sequence. Those bits are used to replace the erroneous bits extracted.

It is possible to have a false positive generated based on the distance of two embedded field labels matching a wrong set of field labels in a structured tree with matching DEF ID bits. But to have a false positive field labeling sequence generated from a structured tree matching the true embedded sequence is almost impossible. It is still up to the user to determine how much/long should the labeling connection sequence resemble the embedded labeling sequence before the obtained bits are considered authentic and match those of the true structured scene.

All the field labels of the extracted PRS bits, i.e. $L2_{protect}[(Field_{(m)Node_{(n)}})_{(k)}]_{detecting}$, $L2_{protect}[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting}$ etc., are saved in the same order as their field DEF ID bits position assigned in the extracted PRS. Those labels will be essential for locating media carrying fields under XMT-A scene structure modification.

Content Location Algorithm

In order to generate a robust algorithm for locating any media carrying fields, the algorithm makes use of the structure signature generated for structure authentication. The media signature and the structure signature are first decrypted with their own private key, as shown in Figure 4-8, and the algorithm is then applied as follows [105]:

(1) The labeling methods $L1(Node_{(n)})$ and $L1(Field_{(m)Node(n)})$ (equations 4-1 and 4-2) are first applied to all nodes and fields of the suspected XMT-A scene structure.

(2) The labeling method $L2_{content\text{-}locate}(Field_{(m)Node(n)})$ (equation 4-4) is then applied to "all" the extremity fields of the scene structure.

(3) Using the first media carrying field label saved as part of the media signature in the embedding process, at location a for instance, i.e $L2_{content\text{-}}$

$_{locate}[(Field_{(c)Node_{(d)}})_{(a)}]_{embedding}$, the distance is calculated with respect to all the saved labels used for structure authentication, i.e. $L2_{protect}[(Field_{(x)Node_{(y)}})_{(i)}]_{embedding}$, $L2_{protect}[(Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding}$, $L2_{protect}[(Field_{(x'')Node_{(y'')}})_{(n)}]_{embedding}$.

(4) The algorithm then tries to find an extremity field label in the scene structure $L2_{content\text{-}locate}[(Field_{(g)N(h)})_{(k)}]_{detect}$ with distances to most of the corresponding field labels of the extracted PRS $L2_{protect}[(Field_{(m)Node_{(n)}})_{(j)}]_{detecting}$, $L2_{protect}[(Field_{(m')Node_{(n')}})_{(j+1)}]_{detecting}$, $L2_{protect}[(Field_{(m'')Node_{(n'')}})_{(p)}]_{detecting}$ that matches that of the media field label in the embedding process $L2_{content\text{-}locate}[(Field_{(c)Node_{(d)}})_{(a)}]_{embedding}$. It should be noted that there is only one media carrying field in the detection phase with distances to all/most of the field labels corresponding to the bits of the extracted PRS that matches a specific media carrying field in the embedding phase with distances to all the labels used to generate the signature for structure authentication. There is almost no chance to have a false positive media carrying field detected. The user can eliminate that chance altogether by defining the minimum number of matching distances with the field labels corresponding to the bits of the extracted PRS that matches a specific media carrying field in the embedding phase with distances to all the labels used to generate the signature for structure authentication, before it is considered as the true expected media carrying field.

(5) The process is repeated to locate all other media carrying fields in the structure.

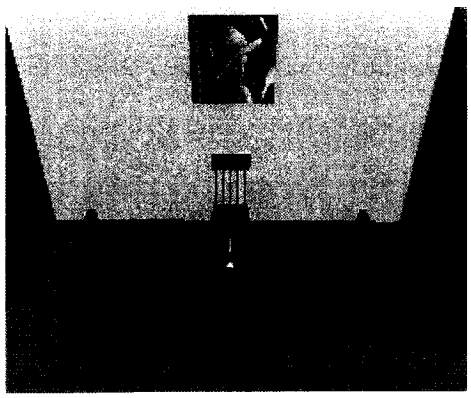
4.3. Experimental Results

Some experiments were conducted to reveal the effectiveness of the proposed algorithm for authenticating a suspicious MPEG-4 scene, locating any watermarked media data located within the scene structure and extracting that watermark to prove ownership, using a common secret generated PRS code.

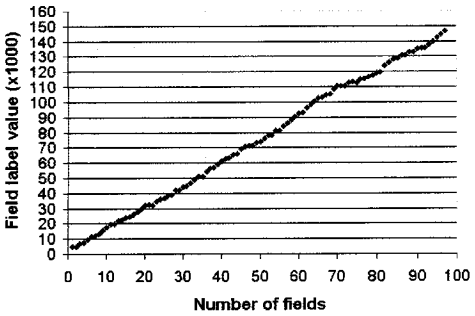
4.3.1. Test Case One

Figure 4-11(a) displays the original MPEG-4 scene used for the testing experiments. The scene contains 322 nodes and 600 fields. The PRS used for scene structure authentication, media content location and watermarking, was generated using the polynomial $h(x) = x^7+x^5+x^4+x^3+1$ of degree m = 7, maximum length PRS of $n = q^m-1 = 2^7-1 = 127$ bits, and with a binary seed value = 0101100. The values of both α and β used for structure field labeling were both set to 1. From that given scene and the chosen PRS, 118 PRS bits were mapped to 97 fields located throughout the XMT-A scene structure. Figure 4-11(b) shows the number of fields used for structure authentication and their labeling values (obtained using equation 4-3), which should remain intact and maintain the same labeling distance with respect to each other for the scene to be considered authentic. Three media data were selected from the given scene; two texture images and one portrait of Lena shown hanging on the wall as displayed in Figure 4-11(a) that were watermarked using the same PRS used for structure authentication. From Figure 4-11(c), it can be seen that the three media carrying fields (label values of 10073, 66248, and 140429), with the field containing the Lena portrait having a labeling value of 140429, were strongly detected with respect to all the 97 selected fields used for scene structure authentication. Having the secret PRS with the appropriate seed value of 44 (0101100 in binary), the embedded watermark in the Lena portrait was strongly detected (shown in Figure 4-11(e)), which can then be proven the ownership of the content and being part of that given MPEG-4 scene even if it was removed from that specific scene and used elsewhere. The watermarked media content detection and watermark extraction algorithm is explained in details in chapter 6.

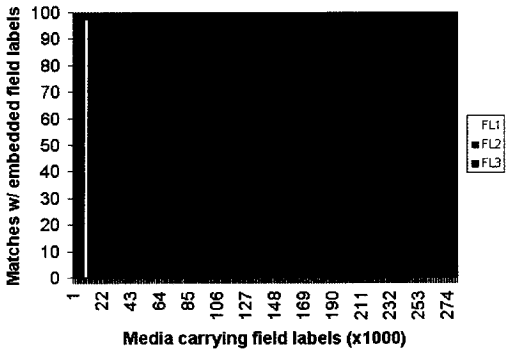
The following are some test results applied on the MPEG-4 XMT structure and the media content to prove the robustness of the algorithm to scene structural tampering and content modification attacks. Table 4-1 displays some of the tampering attacks on the MPEG-4 XMT scene structure, shown in Figure 4-11, and their effect on the scene detection and authentication.



(a) Original dining room

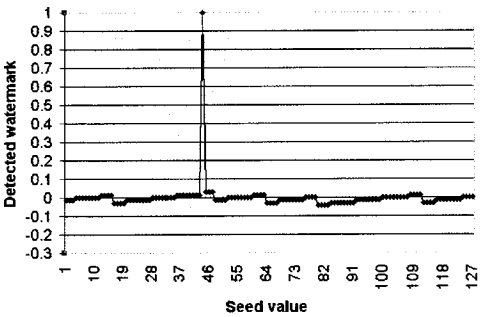


(b) Detected field labels for the original dining room



(c) Located media carrying fields

(d) Original media content



(e) Detected watermark

Figure 4-11. Results from the original MPEG-4 scene protection algorithm

Structure modification	Nodes/Fields present after modification	Structure labeling modified	# of PRS bits recovered (no error correction)	# of PRS bits recovered (error correction)
Original (Figure 4-11(a))	322 Nds / 600 Flds	0 %	118 bits	118 bits
Top shifting (Figure 4-12)	324 Nds / 602 Flds	100 %	0 bits	117 bits
Inner shifting (Figure 4-12)	324 Nds / 602 Flds	16 %	101 bits	117 bits
Deformation (Figure 4-15)	322 Nds / 600 Flds	0 %	118 bits	118 bits
Cropping (Figure 4-17)	209 Nds / 375 Flds	12 %	74 bits	100 bits
Dif. dining room (Figure 4-19(a))	305 Nds / 622 Flds	N/A	0 bits	0 bits
Different scene (Figure 4-20(a))	260 Nds / 520 Flds	N/A	0 bits	0 bits
Diff. Scene w/ orig. (Figure 4-21)	582 Nds / 1120 Flds	100 %	0 bits	117 bits
Topshift+Diff.Scene+Topshift (Figure 4-23)	903 Nds / 1719 Flds	100 %	117 bits	117 bits
Crop+Diff.Scene+Innershift (Figure 4-25)	792 Nds / 1496 Flds	N/A	74 bits (max)	117 bits (max)
BIFS ← → XMT-A	322 Nds / 600 Flds	0 %	118 bits	118 bits
VRML ← → XMT-A	322 Nds / 600 Flds	0 %	118 bits	118 bits

Table 4-1. Effect of tampering attacks on an MPEG-4 dining set scene structure

4.3.1.1. Scene Structure Shifting

Shifting the structure from the top, by adding two extra nodes and two extra fields, did not change anything in the scene view shown in Figure 4-12 from that of the original scene. But by doing this, the structure signature labels did not match with the corresponding labels in the modified structure. Therefore 0 PRS where detected as shown in Table 4-1 and the scene could not be recognized. But after applying the error correction algorithm, explained in section 4.2.1.3 subsection "Scene Structure Detection/Authentication Algorithm", 117 bits of the 118 bits were detected as shown in Table 4-1. From Figure 4-13(a), it can be seen that 96 of the field labels have been detected and maintained the connection sequence with respect to each other, even though the structure has been entirely shifted. Moreover, even though the three media carrying fields have been shifted in the structure position and their labeling values changed to 11093, 67268, and 141449, it can be seen from Figure 4-13(b) that they have been strongly detected with respect to the 96 field labels corresponding to the 117 bits of the detected PRS.

Instead of shifting the whole scene by adding extra nodes and fields from the top and keeping the structure intact, two extra nodes and two extra fields are added within the scene. By doing this, part of the scene structure has shifted in position from the rest of the

scene structure even though the scene view remained the same as that of the original one. From Table 4-1, it can be seen that 16 % of the original structure labeling has changed and the labeling connection sequence has changed from that of the original, starting from field number 81 shown in Figure 4-14(a). The number of detected PRS bits was 101, but after error correction the remaining shifted fields were recognized and 117 PRS bits were recovered. All three media carrying fields were detected, as revealed in Figure 4-14(b), but the media field labeled FL3, was less strongly detected due to its location in the shifted part of the structure, which affected its field labeling connection sequence with respect to the field labels on the non-shifted part of the structure. No other field has been mistakenly detected to be one of those three media carrying fields.

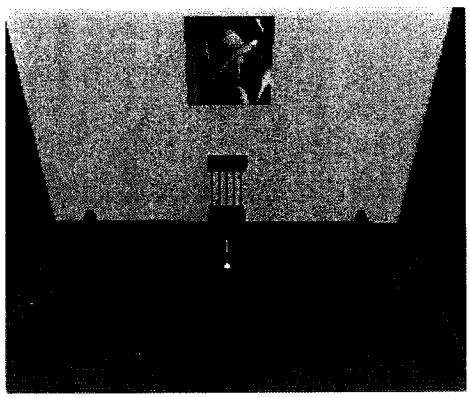
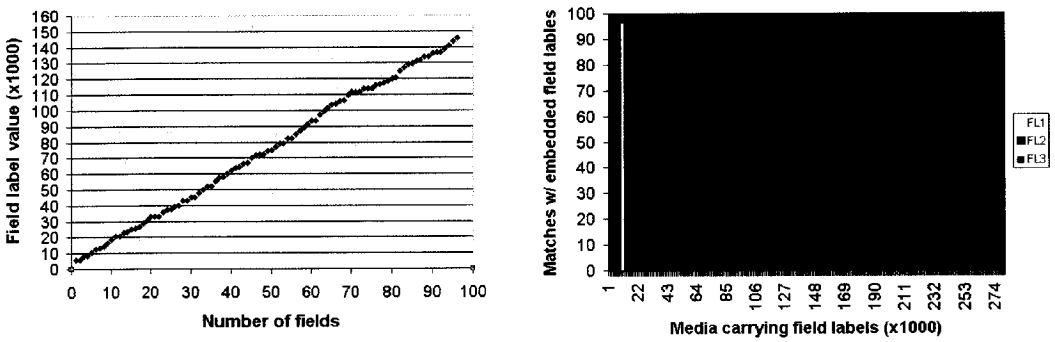


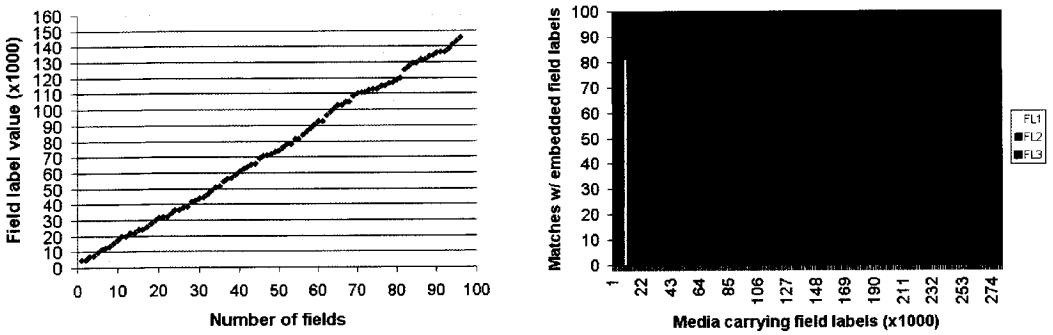
Figure 4-12. Shifted scene



(a) Detected field labels for top shifted scene structure

(b) Located media carrying fields

Figure 4-13. Detected/authenticated original scene and located media content after top shifting XMT-A scene structure



(a) Detected field labels for inner shifted scene structure

(b) Located media carrying fields

Figure 4-14. Detected/authenticated original scene and located media content after inner shifting XMT-A scene structure

4.3.1.2. Scene Deformation

The scene has been modified by adjusting the size and shape of some parts of the scene to make it look different and unnoticeable from the original scene, as displayed in Figure 4-15. The fields' labeling in the structure was not modified as shown in Table 4-1. From Figure 4-16(a), it can be seen that the labeling sequence remained intact and exactly matching that of the original scene. All the media carrying fields were strongly detected as shown in Figure 4-16(b). The protection algorithm is not affected by scaling, rotation, transformation, etc. applied to the objects in the scene. It should also be noted that the algorithm is not affected by swapping of fields within the same node either to change their position within the structure and the scene can still be strongly detected and recognized.

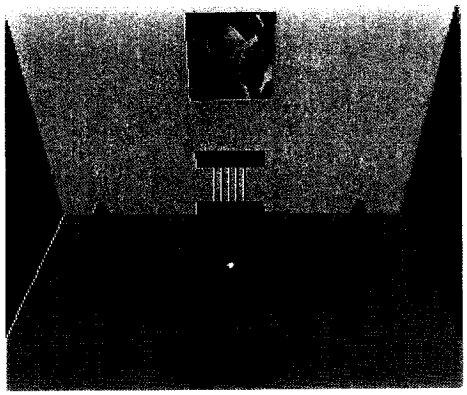
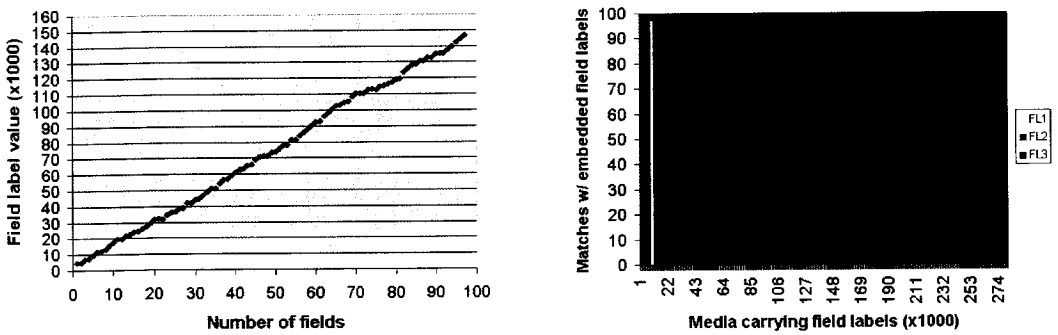


Figure 4-15. Deformed scene



(a) Detected field labels for deformed scene

(b) Located media carrying fields

Figure 4-16. Detected/authenticated original scene and located media content after scene deformation

4.3.1.3. Scene Structure Cropping

The original scene was modified by removing some parts from the end of the XMT-A scene structure, such as the walls, the portrait, the candle and one of the chairs as shown in Figure 4-17. As it can be seen from Figure 4-18(a), the remaining section of the scene was still recognized as being part of the original one even though some of the furniture located in the original scene were removed. Moreover, it is also shown from Figure 4-18(a) that being provided with the original structure signature, the algorithm was able to find a chair in the given scene that matches exactly the deleted chair with a high probability even though it is not located at the original expected part of the structure and

that is why the broken section of the field labeling sequence was generated. The information about the other deleted objects was lost and could not be detected. From this, one can be assured that the scene is the same as the original, even though some parts of the scene, which are the same as other parts of the original scene (chair) have been removed, and some other parts (candle, etc) have been removed and no other parts in the scene match with it. From Figure 4-18(b), it can be seen that only two media carrying fields were detected, because the third media carrying field FL3 containing the Lena portrait was removed with the cropped part of the structure. It can also be observed that the detection signal for the two media carrying fields is not as strong as in the original scene, caused by the removed part of the scene and the loss of some of the labeled fields located in the original scene structure.

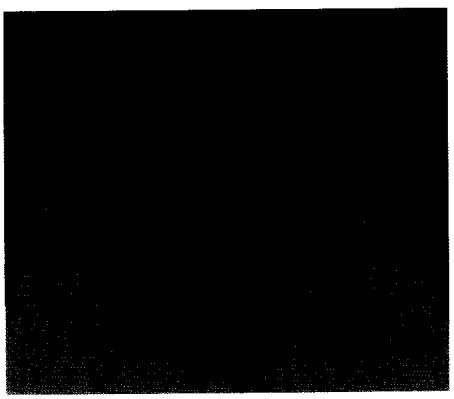
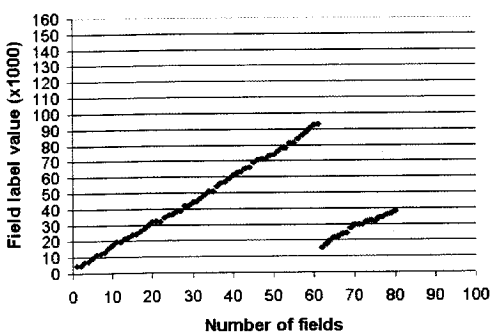
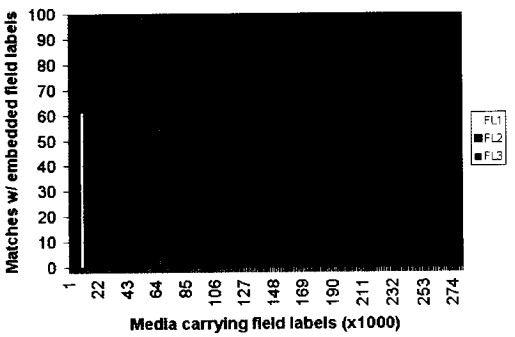


Figure 4-17. Cropped scene



(a) Detected field labels for cropped scene

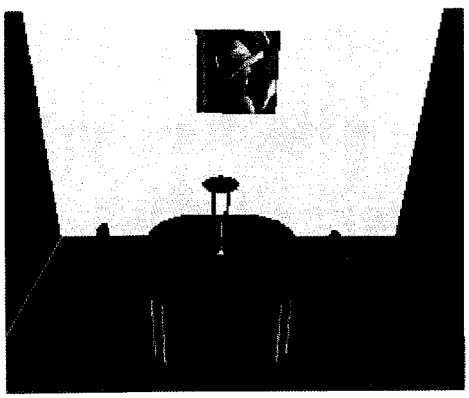


(b) Located media carrying fields

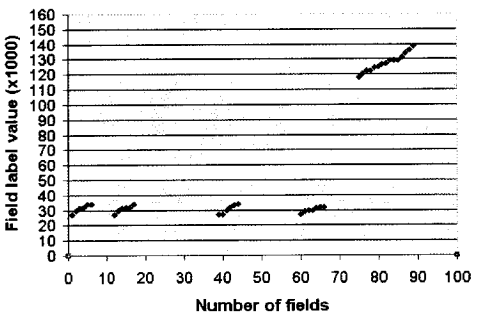
Figure 4-18. Detected/authenticated original scene and located media content after scene cropping

4.3.1.4. Similar Scene

A test was made by having different dining room but maintaining the same number of elements (one table, four chairs, candle, etc), displayed in Figure 4-19(a). Only the candle remained the same. From Figure 4-19(b), it can be seen that the labeling connection sequence was not maintained long enough, except for the part that represented the candle (between field label 117846 and 138537), which was the same as the one used in the original scene. Some scattered PRS bits were detected as a sign of similarity between the given scene and the original one, but the labeling connection sequence was not good enough to maintain those detected bits and after error correction, those bits were rejected as proof of similarity between the two scenes. No media carrying fields were detected in that scene.



(a) Different dining room



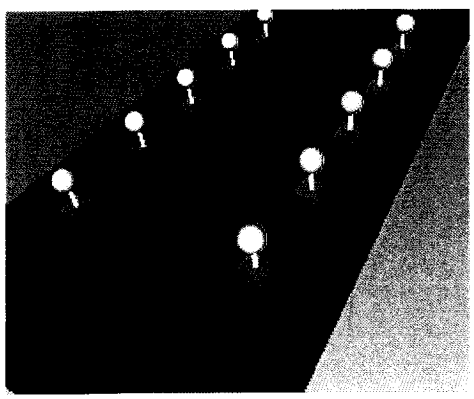
(b) Detected field labels for a different dining room

Figure 4-19. Detected/authenticated original scene when using a different dining room

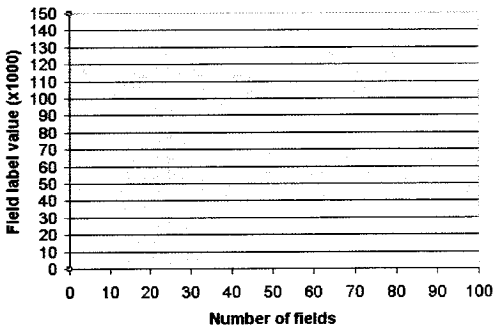
4.3.1.5. A Totally Different Scene with(out) Original Scene

Given a totally different scene, shown in Figure 4-20(a), and being provided with the original signature, no similarity detection was made with the original scene as shown in Figure 4-20(b) and Table 4-1.

The original scene was then added to the end of that given scene structure, combining one scene with another one, as displayed in Figure 4-21. From Table 4-1, it can be seen that the field labeling of the original scene was totally modified. But after applying the algorithm explained in section 4.2.1.3 subsection "Scene Structure Detection/Authentication Algorithm", 117 bits out of the total 118 original PRS bits were detected. From Figure 4-22(a), it can be seen that field labeling sequence remained intact even though the labels were totally modified. From Figure 4-22(b), it can be seen that out of the 1120 fields located in the scene structure, the three specific media carrying fields were correctly detected, even though their field labels were modified and shifted due to the change of location with respect to the whole scene structure.



(a) Totally different scene



(b) Detected field labels for a totally different scene

Figure 4-20. Detected/authenticated original scene when using a totally different scene

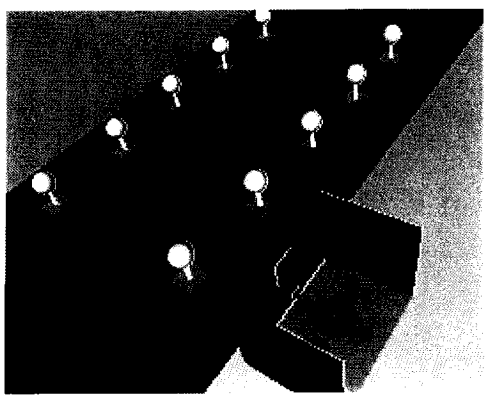
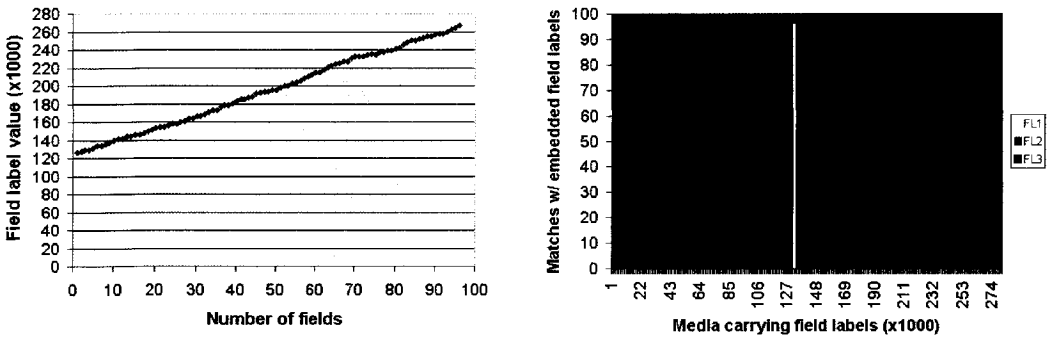


Figure 4-21. Totally different scene combined with the original scene



(a) Detected field labels of diff. w/ original scene combined

(b) Located media carrying fields

Figure 4-22. Detected/authenticated original scene and located media content when combined with a different scene

4.3.1.6. Combining Scenes

A test was made by taking the original scene and placing it multiple times at a different scene as shown in Figure 4-23. One original dining set was placed at the top of the scene structure, and another similar dining set was placed at the end of the scene structure. The whole structure was then shifted by adding a top grouping node. From Table 4-1, it can been seen that the number of nodes and fields almost tripled that of the original scene. From Figure 4-24(a), it can be seen that two original scenes were detected in that different scene, even though they were located at different parts of the scene structure. Each of the media carrying fields was detected twice, as shown in Figure 4-24(b), due to their location in both located dining sets.

The test was taken further, by cropping part of the first original dining set and shifting part of the second dining set by adding an extra node and field inside the dining set scene structure, as displayed in Figure 4-25. From Figure 4-26(a), it can been seen that the algorithm was still able to detect both dining sets, and show the type of modification applied to each one of the original scenes placed in that given scene. From Figure 4-26(b), it can be seen that the media carrying fields were still detected. One of the media

carrying fields was not detected because it was removed with the cropped part. Another media carrying field, located in the second dining set, was less strongly detected than the others due to its location in the inner shifted part of the original dining scene structure.

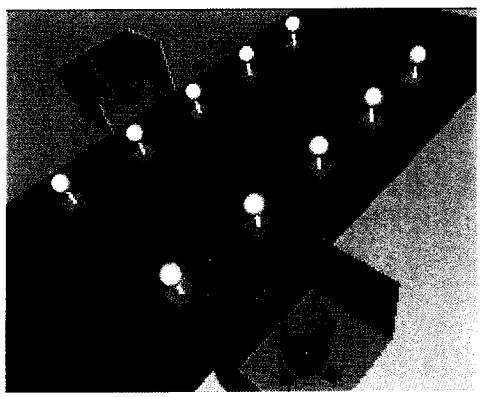
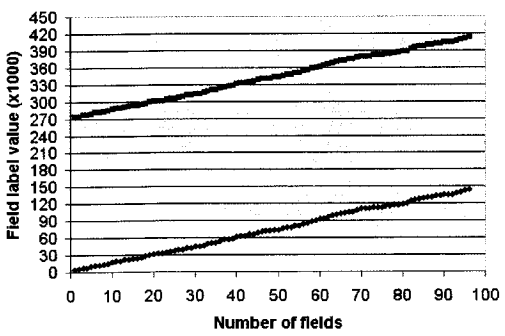
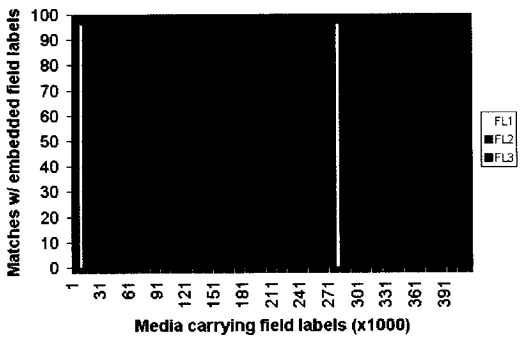


Figure 4-23. An original scene located multiple times in a different scene



(a) Detected field labels of original scene located multiple times in a different scene



(b) Located media carrying fields of original scene located twice in a different scene

Figure 4-24. Detected/authenticated original scene and located media content when inserted multiple times in a different scene

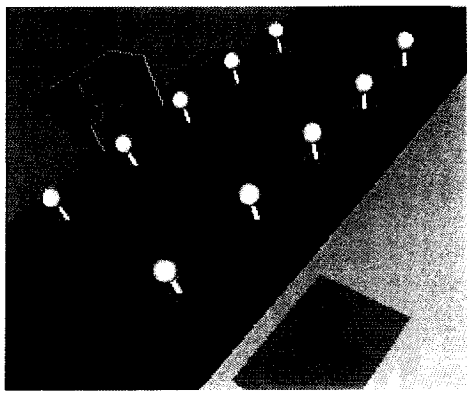
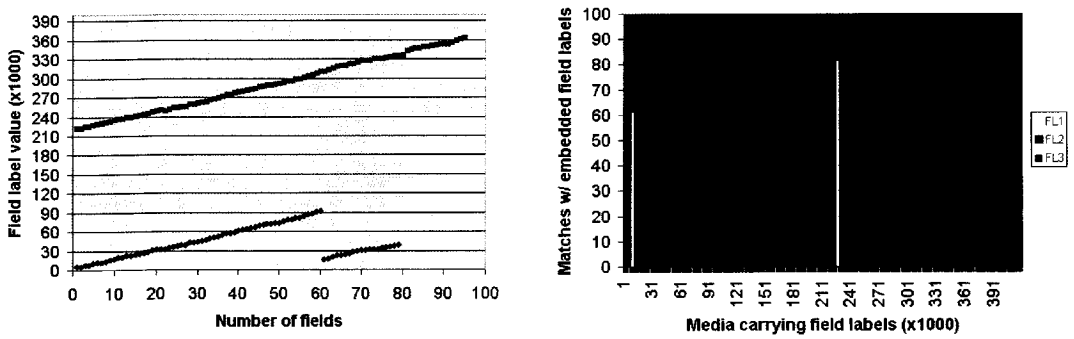


Figure 4-25. Cropped and inner shifted parts of an original scene located in a different scene



(a) Detected field labels of cropped and inner shifted parts of original scene in a different scene

(b) Located media carrying fields

Figure 4-26. Detected/authenticated original scene and located media content when original scene is located twice in a different scene with crop and inner shift applied

4.3.1.7. File Format Conversion

The protection algorithm was also tested for different scene format conversions. The MPEG-4 XMT-A was converted to the MPEG-4 BIFS format and then converted back to XMT-A format. It was also converted to the VRML format and then converted back to XMT-A format. This conversion between one scene format to another had no effect on the scene authentication and media location algorithm as shown in Table 4-1. Therefore, even if the scene is in the MPEG-4 BIFS format, it can still be authenticated by

converting it to the MPEG-4 XMT-A format first before applying the protection algorithm.

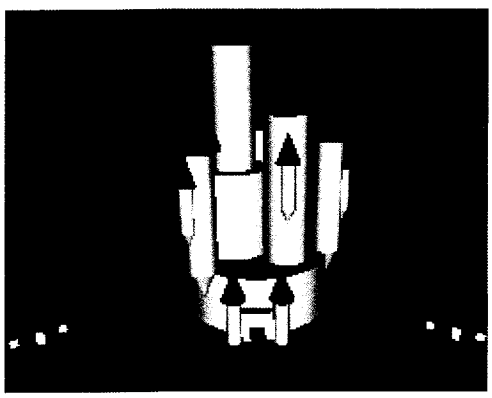
4.3.2. Test Case Two

This second test set is used to check for consistency and accuracy of the algorithm in detecting a given scene under scene structure modifications and show the uniqueness of the generated content signature for a given scene.

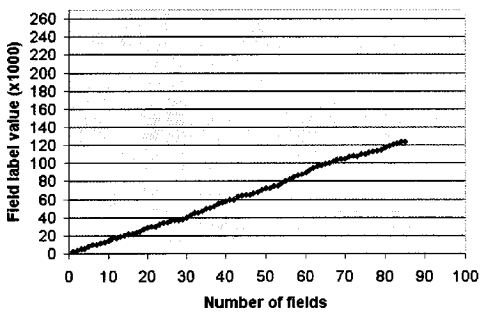
A content signature was generated for the scene shown in Figure 4-27 using the same PRS generated for test case one. The PRS was generated using polynomial $h(x) = x^7+x^5+x^4+x^3+1$, and with a binary seed value = 0101100. The values of both α and β used for structure field labeling are both set to 1. For that given scene and the PRS, 107 PRS bits were mapped to 85 fields located throughout the XMT-A scene structure. Different structural modifications were applied. As it can be seen from Table 4-2, and the results shown in the figures between Figure 4-27 and Figure 4-31, consistency of the output results to those shown in the first test case is revealed. A test was also made by combining the original dining set used in the first test case (Figure 4-11(a)) to the original castle scene, as shown in Figure 4-27(a). Even though both content signatures, one for the original dining set (shown in Figure 4-11(a)), and the other for the original castle signature (shown in Figure 4-27(a)), were both generated using the same polynomial $h(x) = x^7+x^5+x^4+x^3+1$ with the same seed value. When combining both scenes together, and using the content signature generated for the original castle scene in the detection phase, the algorithm was able to detect the castle scene part and not the dining set part of the scene. This shows the uniqueness of the content signature generated for detecting a specific scene.

Structure modification	Nodes/Fields present after modification	Structure labeling modified	# of PRS bits recovered (no error correction)	# of PRS bits recovered (error correction)
Original (Figure 4-27)	269 Nds / 536 Flds	0 %	107 bits	107 bits
Top shifting (Figure 4-28)	272 Nds / 539 Flds	100 %	0 bits	106 bits
Inner shifting (Figure 4-28)	272 Nds / 539 Flds	83 %	18 bits	106 bits
Cropping (Figure 4-30)	227 Nds / 452 Flds	0 %	87 bits	106 bits
Diff. Scene w/ orig. (Figure 4-31)	528 Nds / 1058 Flds	100 %	0 bits	106 bits
Dining set w/ Orig. (Figure 4-32)	590 Nds / 1135 Flds	100 %	0 bits	106 bits

Table 4-2. Effect of tampering attacks on original MPEG-4 castle scene structure



(a) Original castle scene



(b) Detected field labels for the original scene

Figure 4-27. Original castle with its detected field labels

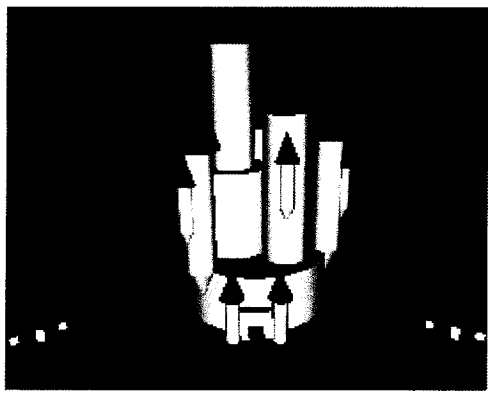
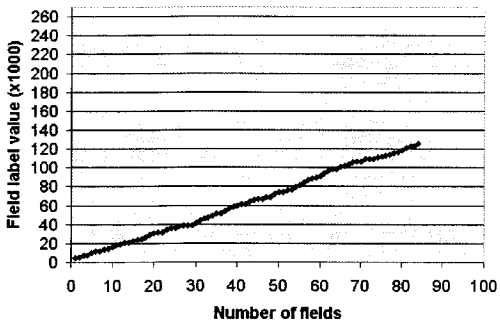
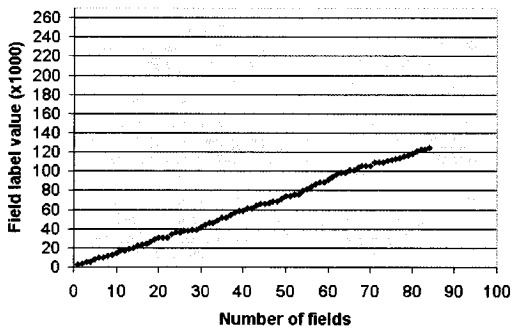


Figure 4-28. Shifted scene structure

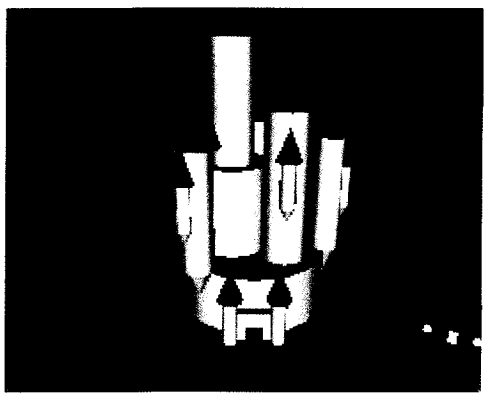


(a) Detected field labels in top shifted structured scene

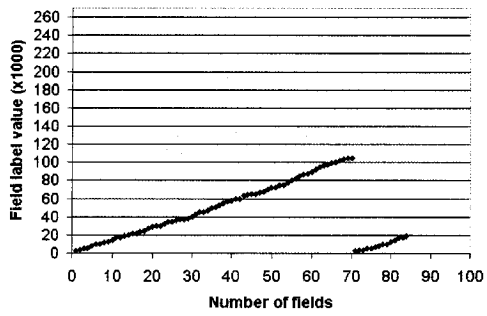


(b) Detected field labels in inner shifted structured scene

Figure 4-29. Detected field labels after applying top and inner shifting to the original scene structure

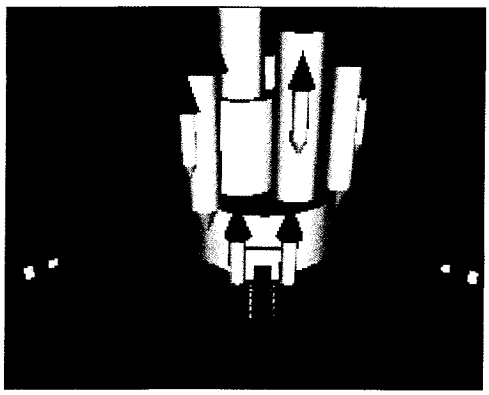


(a) Original scene with cropped part

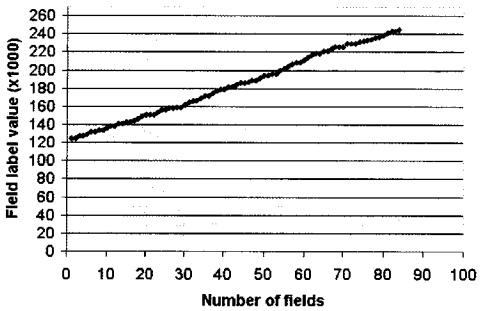


(b) Detected field labels in the cropped scene

Figure 4-30. Original scene with a cropped part and the detected field labels

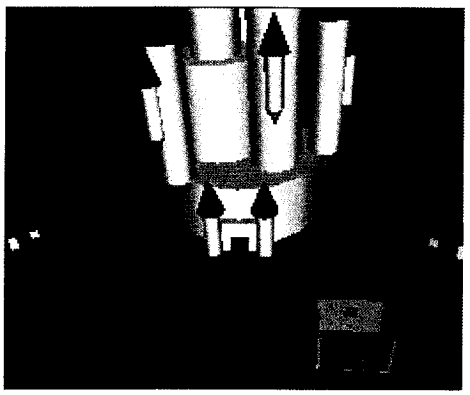


(a) Original scene added to another scene

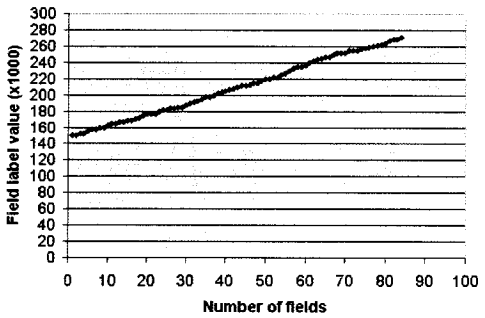


(b) Detected field labels of the original scene located in a different scene

Figure 4-31. Original scene added to a diff. scene with detect field labels shown



(a) Original castle scene combined with the dining set



(b) Detected field labels of the original castle scene

Figure 4-32. Original castle combined with the original dining set and detected field labels shown

Chapter 5

Robust SMIL Content-Based Scene Structure Authentication and Content Protection

This chapter presents a novel algorithm for the protection of SMIL scenes and their media content. The algorithm is a generalization of the one proposed for MPEG-4 scenes (explained in chapter 4). Even though the algorithm has been applied on SMIL scenes, it can also be used for any scenes defined in XML format with predefined number of nodes and attributes.

A PRS is used to generate two content-based signatures of a given SMIL scene. The first signature is used to detect and authenticate a specific scene structure. This generated signature is content fragile and any modification to the structure's content will be detected, yet the algorithm applied with that generated signature is robust against structure modifications and able to detect a scene even after major scene structure tampering/modification. The second generated signature, after applying a similar technique, is used to locate specific nodes that contain the media data and need to be protected. With the same generated PRS code used for the detection of a given SMIL

scene, the media data, i.e. images, are watermarked. The watermarking algorithm for the media data makes use of spread spectrum watermarking proposed by Cox et al. [21]. Even if only part of a scene with its content has been unlawfully used, using the secret generated PRS code, the algorithm can still detect the scene, locate and extract the watermark from the media content and link the SMIL scene with its content to the rightful owner.

To the best of our knowledge, no robust protection algorithm has yet been revealed that focuses on the protection of SMIL scenes and their media content, which is the focus of this chapter.

The following section will give detailed explanation on the proposed algorithm for the protection of SMIL scenes and their content. For general information on SMIL, please refer to the appendix.

5.1. SMIL Proposed Protection Algorithm – Structure and Content

This section explains the proposed robust algorithm for detection and authentication of SMIL scene structures and their media content. It should be noted that the proposed algorithm is designed to withstand major scene structure modifications and still be able to authenticate a given scene and locate the watermarked media content. Some of the modifications on the XML structures would be: splitting the structure into multiple independent usable structured scenes, addition/elimination of insignificant nodes in the scene, and modification of node content within usability limits. In all modifications, the main idea is to try to destroy the structure properties while still preserving the usability of the given scene.

A general architecture of the proposed protection system is displayed in Figure 4-2 and Figure 4-3. With a privately (secret) generated PRS, a unique SMIL scene content-based signature is obtained, shown in Figure 4-2, after passing a specific SMIL scene structure through the structure detection/authentication algorithm (explained below in details), which is kept secret for later detection/authentication of a suspicious SMIL scene

structure. With the same private PRS, the media content (images, audio, video, etc.) located within a SMIL scene that needs to be protected, are passed through the watermarking algorithm to obtain the watermarked media objects. The media content signature is then generated after passing the scene structure through the content location algorithm. With the media content signature, the protection algorithm is able to locate the watermarked media objects in the scene even after major structure modifications.

The detection and authentication system for SMIL scenes is revealed in Figure 4-3. Having a suspicious SMIL scene, with the aid of the secret PRS and the scene structure signature, the structure protection algorithm is then able to detect and authenticate that given scene. If the scene was detected and related to the suspected one, the algorithm then generates a labeling sequence for that given scene. With that labeling sequence when combined with the secret scene structure signature and the media content signature, is able to locate all of the watermarked media content, even if the SMIL scene structure has undergone structural modifications. Having located the watermarked media objects, the watermarking algorithm is then able to detect the embedded PRS watermark and extract it, which is the same PRS used to generate the scene structure signature and media location signature. With this protection algorithm, it is easy to link the SMIL scene structure and its media content to the same true owner.

5.1.1. SMIL scene structure detection and media content location algorithms

This section provides a detailed explanation of the algorithms used for SMIL scene structure detection/authentication and media content location.

5.1.1.1. Labeling

In order to identify and secure SMIL structured scenes, some form of node labeling should be considered. It can provide unique information in identifying and marking certain nodes that can still be detected even after structure modification attacks.

However, considering the fact that the labeling method should withstand graph structure modifications due to tampering attacks and still be able to identify the labeled nodes, would render most of the graph labeling algorithms [2][3][28][33][54][71] useless in that domain.

A solution that would make node labeling more robust to structural changes would be to consider the SMIL scene structure as composed of nodes and their corresponding fields (attributes). Before looking at a specific SMIL scene structure, each SMIL node is assigned a unique type number which will distinguish it from other nodes later on. The fields of every given node are also assigned IDs in increasing order starting from 0 using the minimum number of bits, a method used in the MPEG-4 BIFS compression standard to achieve high compression rate [43][48]. For example, SMIL node "Anchor", shown in Figure 5-1, is assigned a static node type equal to 60 in my algorithm, and its seven fields are assigned the minimum bit IDs in order as follows (id = 000, show = 001, href = 010, skip-content = 011, cords = 100, begin = 101, end = 110). This numbering will be used later on for scene structure detection and authentication.
Now given a specific SMIL scene structure, the nodes are first labeled as follows:

$$\begin{aligned} L1(Node_{(0)}) &= seed + Node_{(0)}Type \\ L1(Node_{(1)}) &= L1(Node_{(0)}) + Node_{(1)}Type + \delta \\ L1(Node_{(n)}) &= L1(Node_{(n-1)}) + Node_{(n)}Type + \delta \end{aligned} \qquad (5\text{-}1)$$

where:

seed: a starting integer value.

n: node number in a SMIL tree.

Adding the SMIL node type to the node's label, would make the label unique and adds more robustness to the scene under structural attacks.

The fields (attributes) are then labeled with respect to their parent's node as follows:

$$L1(Field_{(0)Node_{(n)}}) = L1(Node_{(n)}) + \gamma$$
$$L1(Field_{(m)Node_{(n)}}) = L1(Field_{(m-1)Node_{(n)}}) + \gamma \quad (5\text{-}2)$$

where:

L1: general labeling algorithm used to label each node/field and identify it.

δ, γ: constant integer increment factors.

n: node number in a SMIL tree.

m: field number of a specific node.

As it can be seen, the fields' labels are directly related to their parent's node label and each other's labels and not to other node labels. This will make the algorithm more robust to structural modifications as it will be demonstrated in the next section.

A second labeling algorithm is then applied to a select number of fields depending on whether the fields are used for later scene structure detection/authentication or media content location. This labeling method is generated based on the field's label with respect to its parent node label and above node labels in the scene structure. The second labeling algorithm can be defined as follows:

$$L2(Field_{(m)Node_{(n)}}) = \alpha * L1(Field_{(m)Node_{(n)}}) + \beta * \sum_{x=n-1}^{n} L1(Node_{(x)}) \quad (5\text{-}3)$$

where:

L2: a second labeling algorithm used on selected fields containing media data objects.

α: determines the weight of a certain field – value can range between 0.1 and 1.0.

β: provides more control and stability for a field with respect to its node neighbors – value can range between 0.1 and 1.0.

m,n: field with index m for a given node with index n.

x: index of a node above a specific field.

The values for α and β can be adjusted according to the stability and robustness required. By increasing α for instance, greater labeling stability can be achieved. On the other hand, increasing the value of β makes a field's label more robust with respect to its neighboring nodes. Certain tradeoffs need to be considered on selecting those two values to achieve optimum results.

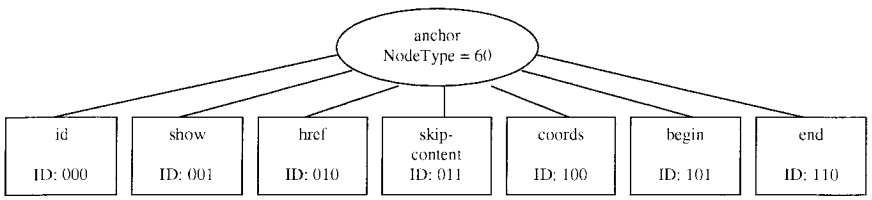


Figure 5-1. Node with its fields

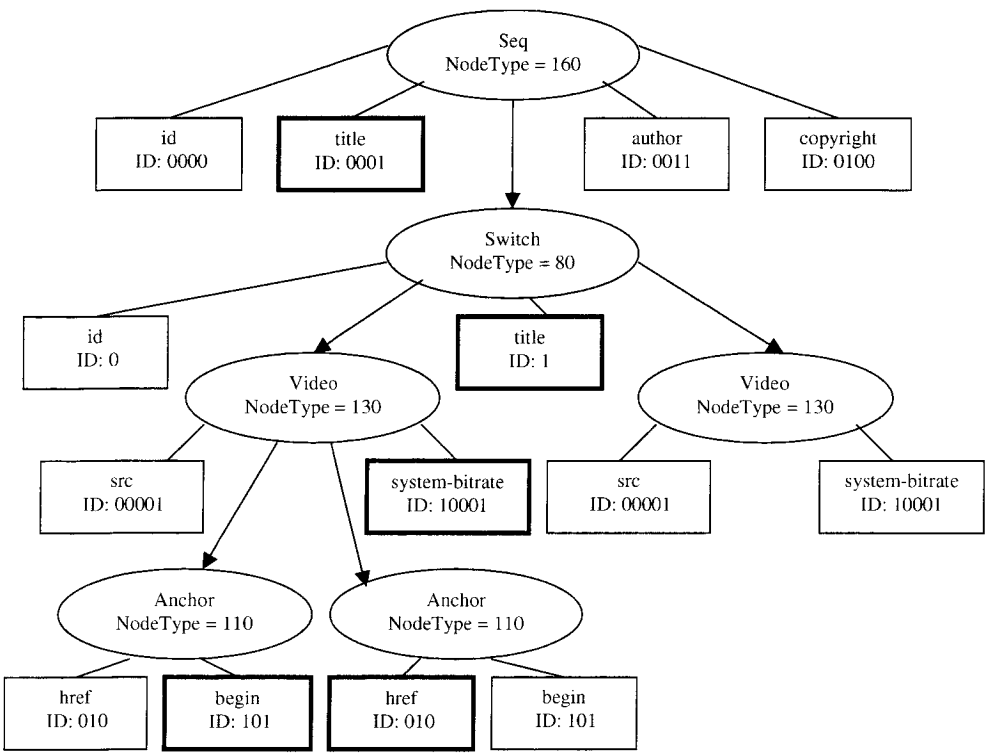


Figure 5-2. Sample of SMIL scene structure labeling

5.1.1.2. Signature Generation

This section deals with the details involved in generating the signatures used for SMIL scene structure authentication and media content location.

Scene Structure Authentication Signature

To be able to authenticate the scene structure and detect any modifications being applied to it, a scene authentication signature is generated. The algorithm makes use of Pseudo-Random Sequences (PRS) encoding.

Given a primitive polynomial *h(x)* of degree *m* with coefficients from a Galois Field with *q* elements GF(*q*) [7][10], a pseudo-random encoded sequence can be generated [4][18][23][31][40][99]. This pseudo-random sequence generated will be used to gather unique information from the SMIL structured scene. For example, if $q = 2$, $m = 4$, $h(x) = x^4 + x^3 + 1$, the following PRS will be generated:

PRS: 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1

Having a window of size *m* slide across that sequence is unique [89]. By mapping this information to data gathered from the structured scene, then any modification to the structure will damage part of this continuous sequence and can therefore be detected and located. Moreover, even if an attack/modification got applied to the scene, the scene will still be detected and recognized.

To map the PRS to the scene structure, the algorithm goes through the SMIL scene structure and selects those fields with IDs (explained in section 5.1.1.1) that correspond to the PRS bits that were generated. If a field cannot be found with an ID that corresponds to the required bits, an extra field corresponding to a specific node can be inserted into the structure that carries required ID value. All the nodes in the structure with their corresponding fields are then labeled using the $L1(Node_{(n)})$ and $L1(Field_{(m)Node_{(n)}})$ equations 5-1 and 5-2 given in section 5.1.1.1. The selected fields are then labeled again using the equation $L2(Field_{(m)Node_{(n)}})$ (equation 5-3), using an

appropriate value for both α and β, where *m* is the index of a chosen field belonging to node *n*. For example, given part of a generated PRS "0001110001101010", information can be gathered from the SMIL scene, shown in Figure 5-2, by selecting the fields {title, title, system-bitrate, begin, href} with IDs {0001, 1, 10001, 101, 010} respectively. Not every node along the path needs to have a field that is labeled for structure identification. The labels generated $L2(Field_{(m)Node_{(n)}})_{(i)}$ and the number of bits matches to some bits in the PRS in each field - $BitsNum(Field_{(m)Node_{(n)}})_{(i)}$, where *i* is the index of the saved fields Labels/BitsNums – are then saved. This will be the structured scene unique signature for later identification/authentication. The algorithm can be applied to different parts of the structure to increase the robustness of the structure against different kinds of attacks. The signature generated and the PRS sequence should remain secret and each can be encrypted with a symmetric secret key, as shown in Figure 4-6, for later detection/authentication of a given scene if required.

Content Location Signature

An algorithm similar to the one used for scene structure detection is used for media content location. All the nodes in the structure with their corresponding fields are labeled using the $L1(Node_{(n)})$ and $L1(Field_{(m)Node_{(n)}})$ (equations 5-1 and 5-2), as done for the authentication signature generation. The media carrying fields are then labeled again using the equation 5-3, $L2(Field_{(x)Node_{(y)}})$, using an appropriate value for both α and β, where *x* is the index of a chosen field belonging to node *y*. All the labels generated $L2(Field_{(x)Node_{(y)}})_{(i)}$ will be the saved media signature (with each label at a specific index *i*) and encrypted if necessary, as shown in Figure 4-6, for later detection of those specific media carrying fields.

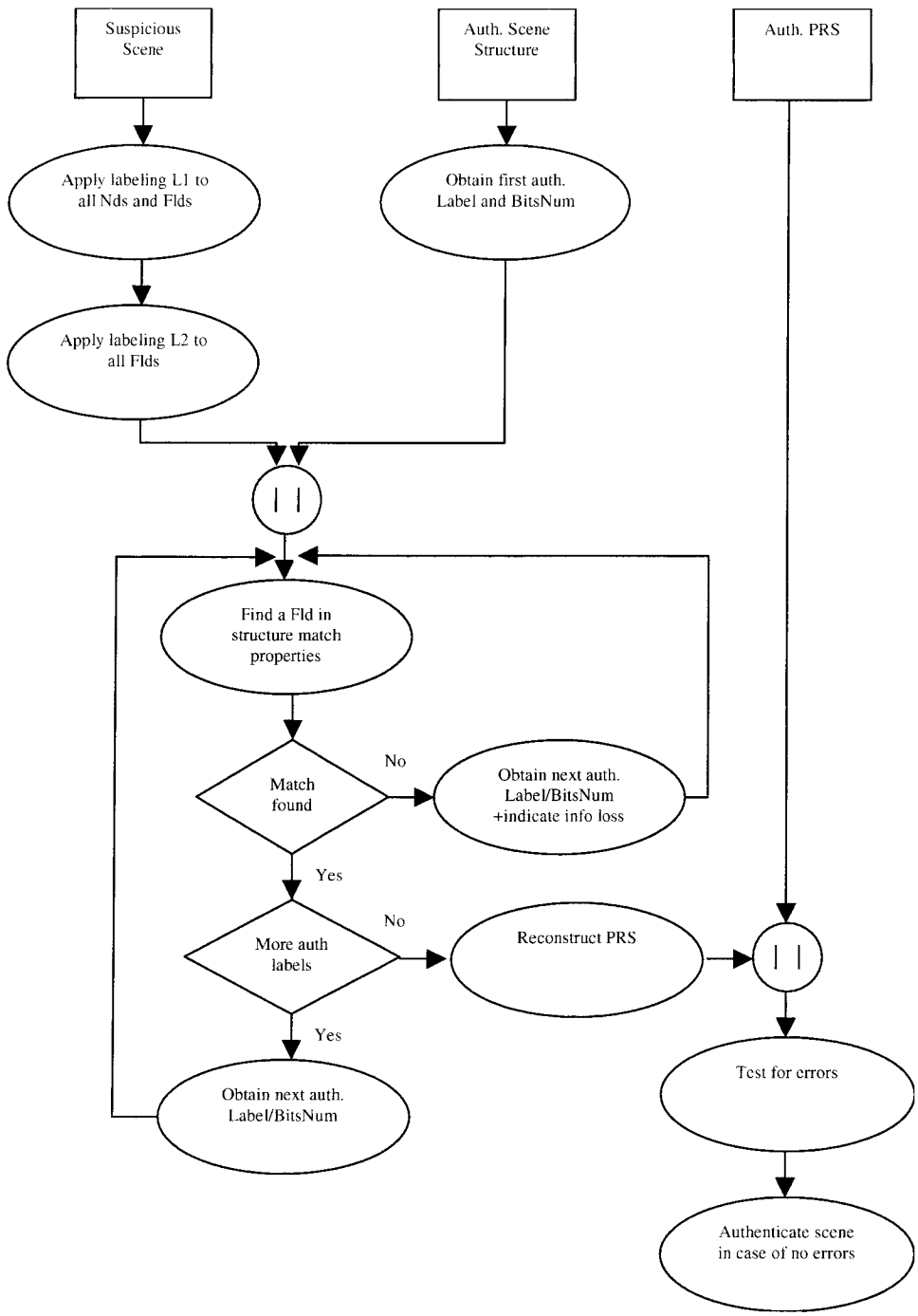


Figure 5-3 Detection/Authentication algorithm for SMIL scene structures.

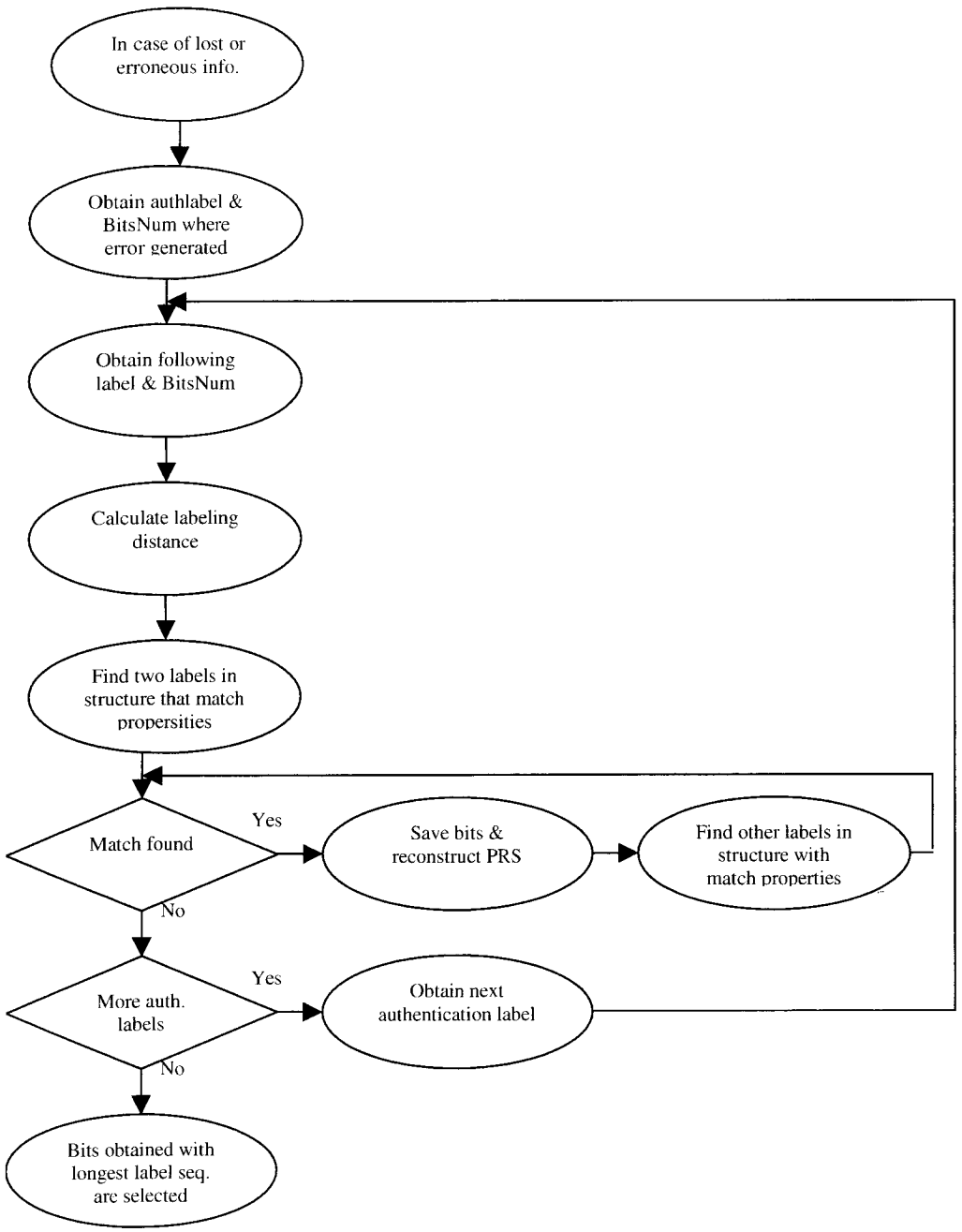


Figure 5-4. Detection/Authentication algorithm for SMIL scene structures with erroneous or missing data.

5.1.1.3. Scene Detection and Content Location

In the event of suspicious unauthorized use of a SMIL scene, the following algorithm is applied.

Scene Structure Detection/Authentication Algorithm

Given a suspicious scene, the structure signature is decrypted with the secret key and with the suspected scene structure a PRS is generated, as shown in Figure 4-7. With the second private key, the secret PRS is decrypted and compared to the generated PRS. The comparison will determine whether this suspected scene is the original copyrighted scene or not. General flowcharts of the detection and identification algorithm of a scene are shown in Figure 5-3 and Figure 5-4. A detailed explanation of the algorithm is as follows (it should be noted that the word "embedding" is used for the fields selected and used in the signature generation process, and the words "extracting" or "detecting" are used for fields selected and used from the suspected scene structure in the detection/identification process):

(1) Obtain the suspicious structured scene.

(2) Apply the labeling methods $L1(Node_{(n)})$ and $L1(Field_{(m)Node_{(n)}})$ (equations 5-1 and 5-2) to all nodes and fields in the SMIL scene structure.

(3) Apply the labeling method $L2(Field_{(m)Node_{(n)}})$ (equation 5-3) to "all" the fields of the scene structure.

(4) Compare all the labeled field values to their matching labeled field value selected during embedding (field selection process), $L2[(Field_{(m)Node_{(n)}})]_{detecting} = L2[(Field_{(x)Node_{(y)}})_{(i)}]_{embedding}$. It should be noted that m is a specific field of a given node n in the suspicious scene structure, and x is a specific field of a given node y found at location i of the structure signature.

(5) If there is a match between the embedded field label and the extracted field label, then check if the number of bits of the field ID in the extracted field matches with the

number of bits in the field used for embedding $BitsNum[(Field_{(m)Node_{(n)}})]_{detecting} = BitsNum[(Field_{(x)Node_{(y)}})(i)]_{embedding}$

(6) If there is a match, the field ID bits are assigned for reconstructing the m-sequence. If there is no match, an error is detected.

(7) The primitive polynomial $h(x)$ of degree m with coefficients from $GF(q)$, used to generate the PRS in the embedding process, is now used to check for the correctness of the extracted sequence and locate the erroneous data.

In the case of erroneous or missing bits from the extracted sequence (caused due to structure tampering/modification), an error correction/elimination process is done by applying an exhaustive search to the scene structure provided to locate the erroneous/missing data.

(8) Obtain the $L2[(Field_{(x)Node_{(y)}})(i)]_{embedding}$ and $BitsNum[Field_{(x)Node_{(y)}})(i)]_{embedding}$ where the erroneous/missing bits were detected in the extracted sequence.

(9) Obtain the next label and bit number of the field used for PRS embedding, $L2[Field_{(x')Node_{(y')}})(i+1)]_{embedding}$ and $BitsNum[Field_{(x')Node_{(y')}})(i+1)]_{embedding}$, where field x' of node y' indicate a different field located at index $i+1$ of the structure signature.

(10) Calculate the distance between the two labels; Distance = $L2[(Field_{(x')Node_{(y')}})(i+1)]_{embedding} - L2[(Field_{(x)Node_{(y)}})(i)]_{embedding}$.

(11) Given the bit number of the two consecutive fields used for sequence embedding, $BitsNum[(Field_{(x)Node_{(y)}})(i)]_{embedding}$ and $BitsNum[Field_{(x')Node_{(y')}})(i+1)]_{embedding}$ the expected/correct sequence bits can be obtained from the generating polynomial $h(x)$.

(12) An exhaustive search is made through the scene structure trying to find two fields with labeling distance that matches the distance of the labeled fields used for embedding; $L2[(Field_{(m')Node_{(n')}})(k+1)]_{detecting} - L2[(Field_{(m)Node_{(n)}})(k)]_{detecting} = L2[Field_{(x')Node_{(y')}})(i+1)]_{embedding} - L2[Field_{(x)Node_{(y)}})(i)]_{embedding}$. It should be noted that

indices k and $k+1$ in the detecting process indicate the different structure location of two different fields m and m', whereas indices i and $i+1$ in the embedding process indicate the fields x and x' location in the structure signature.

(13) If there is a match in the labeling distance, check the field ID of those two detected fields with the expected bits obtained from the generating polynomial using the $BitsNum[(Field_{(x)Node_{(y)}})_{(i)}]_{embedding}$ and $BitsNum[Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding}$

(14) If there is a match, save the bits obtained in a memory location $\text{Mem1(Rbits)}_{(a)}$ and save $L2[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting}$ in a different memory location $\text{Mem2}(L2[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting})_{(a)}$, where a is the index of the memory location.

(15) Another search is made through the structure to find another two fields with labeling distance that matches the distance of the labeled fields used for embedding and field IDs that match those from the polynomial sequence (repeat from step12); $L2[(Field_{(m')Node_{(n')}})_{(p+1)}]_{detecting} - L2[(Field_{(m)Node_{(n)}})_{(p)}]_{detecting} = L2[Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding} - L2[Field_{(x)Node_{(y)}})_{(i)}]_{embedding}$

(16) If there is a match, save the bits obtained in the next position in the memory $\text{Mem1(Rbits)}_{(a+1)}$ and $L2[(Field_{(m')Node_{(n')}})_{(p+1)}]_{detecting}$ in $\text{Mem2}(L2[(Field_{(m')Node_{(n')}})_{(p+1)}]_{detecting})_{(a+1)}$

(17) The search continues until there are no more matches.

(18) The distance between the next two embedded field labels is considered; $L2[Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding}$ and $L2[Field_{(x)Node_{(y)}})_{(i+2)}]_{embedding}$, and the same process is repeated starting from step 8.

(19) If there is a match between $L2[(Field_{(m)Node_{(n)}})_{(k+2)}]_{detecting} - L2[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting} = L2[Field_{(x)Node_{(y)}})_{(i+2)}]_{embedding} - L2[Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding}$, and the field IDs with the expected polynomial sequence bits, then search for the field label $L2[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting}$ in the memory $\text{Mem2}(L2[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting})$. If the label is found in the memory,

i.e. $Mem2(L2[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting})_{(a)}$, and is directly related with other detected labels, then replace the field label by the next field label $Mem2(L2[(Field_{(m)Node_{(n)}})_{(k+2)}]_{detecting})_{(a)}$ and concatenate the bits obtained to the previous ones located in $Mem1(Rbits)_{(a)}$.

(20) The search continues for finding matching distances between field labels in the structured tree and the embedded field labels.

(21) When there are no more matches, the distance between the next two embedded fields is considered $L2[Field_{(x)Node_{(y)}})_{(i+2)}]_{embedding}$ and $L2[Field_{(x')Node_{(y')}})_{(i+3)}]_{embedding}$, and the process is repeated until there are no more consecutive embedded fields to consider.

(22) The concatenated bits found in Mem1(Rbits) with the longest corresponding labeling sequence located in Mem2, are considered the most authentic and matching to the true embedded sequence, since the consecutive labeling sequence of them matches those of the embedded sequence. Those bits are used to replace the erroneous bits extracted.

It is possible to have a false positive generated based on the distance of two embedded field labels matching a wrong set of field labels in a structured tree with matching field ID bits. But to have a false positive field labeling sequence generated from a structured tree matching the true embedded sequence is almost impossible. It is still up to the user to determine how much/long should the labeling connection sequence resemble the embedded labeling sequence before the obtained bits are considered authentic and match those of the true structured scene.

All the field labels of the extracted PRS bits, i.e. $L2[(Field_{(m)Node_{(n)}})_{(k)}]_{detecting}$, $L2[(Field_{(m')Node_{(n')}})_{(k+1)}]_{detecting}$ etc., are saved in the same order as their field ID bits position assigned in the extracted PRS. Those labels will be essential for locating media carrying fields under the SMIL scene structure modification.

Content Location Algorithm

In order to generate a robust algorithm for locating any media carrying fields, the algorithm makes use of the structure signature generated for structure authentication. The media signature and the structure signature are first decrypted with their own private key, as shown in Figure 4-8, and the following algorithm is then applied:

(1) The labeling methods $L1(Node_{(n)})$ and $L1(Field_{(m)Node(n)})$ (equations 5-1 and 5-2) are first applied to all nodes and fields of the suspected SMIL scene structure.

(2) The labeling method $L2(Field_{(m)Node(n)})$ (equation 5-3) is then applied to "all" the fields of the scene structure.

(3) Using the first media carrying field label saved as part of the media signature in the embedding process, at location a for instance, i.e $L2[(Field_{(c)Node_{(d)}})_{(a)}]_{embedding}$, the distance is calculated with respect to all the saved labels used for structure authentication, i.e. $L2[(Field_{(x)Node_{(y)}})_{(i)}]_{embedding}$, $L2_{protect}[(Field_{(x')Node_{(y')}})_{(i+1)}]_{embedding}$, $L2_{protect}[(Field_{(x'')Node_{(y'')}})_{(n)}]_{embedding}$.

(4) The algorithm then tries to find a field label in the scene structure $L2[(Field_{(g)N(h)})_{(k)}]_{detect}$ with distances to most of the corresponding field labels of the extracted PRS $L2[(Field_{(m)Node_{(n)}})_{(j)}]_{detecting}$, $L2[(Field_{(m')Node_{(n')}})_{(j+1)}]_{detecting}$, $L2[(Field_{(m'')Node_{(n'')}})_{(p)}]_{detecting}$ that matches that of the media field label in the embedding process $L2[(Field_{(c)Node_{(d)}})_{(a)}]_{embedding}$. It should be noted that there is only one media carrying field in the detection phase with distances to all/most of the field labels corresponding to the bits of the extracted PRS that matches a specific media carrying field in the embedding phase with distances to all the labels used to generate the signature for structure authentication. There is almost no chance to have a false positive media carrying field detected. The user can eliminate that chance altogether by defining the minimum number of matching distances with the field labels corresponding to the bits of the extracted PRS that matches a specific media carrying

field in the embedding phase with distances to all the labels used to generate the signature for structure authentication, before it is considered as the true expected media carrying field.

(5) The process is repeated to locate all other media carrying fields in the structure.

5.2. Experimental Results

Some experimental results were obtained, in order to reveal the effectiveness of the proposed algorithm for detecting and authenticating a suspicious SMIL scene, locating any watermarked media data located within the scene structure and extracting that watermark to prove ownership, using a common secret generated PRS code.

5.2.1. Test Case One

Figure 5-5(a) shows a snapshot of the original multimedia SMIL scene used for the testing experiments. The whole scene contains 89 nodes and 303 fields. The PRS used for scene structure authentication, media content location and watermarking, was generated using the polynomial $h(x) = x^7+x^5+x^4+x^3+1$ of degree m = 7, maximum length PRS of $n = q^m-1 = 2^7-1 = 127$ bits, and with a binary seed value = 0101100. The values of both α and β used for structure field labeling were both set to 1. From that given scene and the chosen PRS, 121 PRS bits were mapped to 48 fields located throughout the SMIL scene structure. Figure 5-5(b) shows the number of fields used for structure detection/authentication and their labeling values, which should remain intact and maintain the same labeling distance with respect to each other for the scene to be considered authentic. Four images were selected from the given scene; with the image of Lena shown in Figure 5-5(a) being one of those, that were watermarked using the same PRS used for structure authentication. From Figure 5-5(c), it can be seen that the four media carrying fields, with the field containing the Lena portrait (labeled "FL1") having a labeling value of 15878, were strongly detected with respect to all the 48 selected fields used for scene structure authentication. Having the secret PRS with the appropriate seed

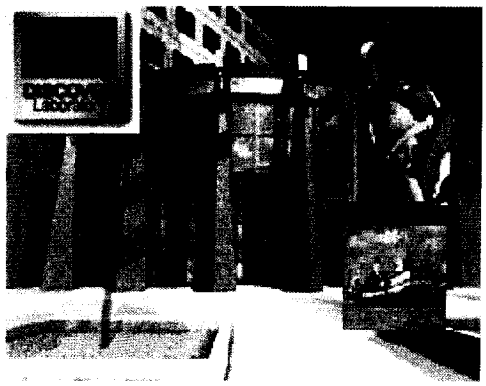
value of 44 (0101100 in binary), the embedded watermark in the Lena image was strongly detected (shown in Figure 5-5(e)), which can then be proven the ownership of the content and being part of that given SMIL scene even if it was removed from that specific scene and used elsewhere. The media content protection will be explained in details in chapter 6.

The following are some test results applied on the SMIL structure to prove the robustness of the algorithm to scene structural tampering attacks.

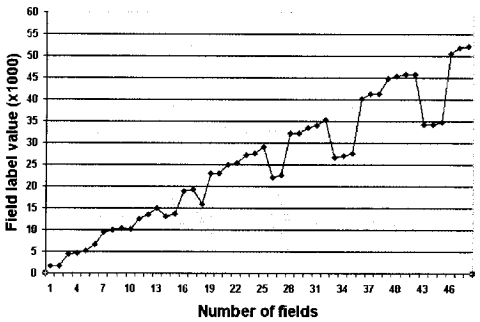
Table 5-1 displays some of the tampering attacks on the original SMIL scene, shown in Figure 5-5, and their effect on the scene detection and authentication.

5.2.1.1. Scene Structure Shifting

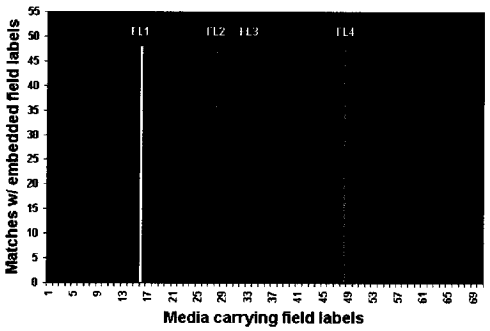
Shifting the structure from the top, by adding two extra neutral nodes and four extra fields, did not change anything in the scene view/presentation from that of the original scene. But by shifting the scene structure, the original signature labels did not match with the corresponding labels in the modified structure. Therefore 0 PRS where detected as shown in Table 5-1, and the scene could not be recognized. But after applying the error correction algorithm, explained in section 5.1.1.3 subsection "Scene Structure Detection/Authentication Algorithm", 120 bits of the 121 bits were detected as shown in Table 5-1. From Figure 5-6(a), it can be seen that 47 of the field labels have been detected and maintained the connection sequence with respect to each other, even though the structure has been entirely shifted. The media carrying fields, shown in Figure 5-6(b), have been strongly detected with respect to the 47 field labels corresponding to the 120 bits of the detected PRS.



(a) Section of original SMIL scene

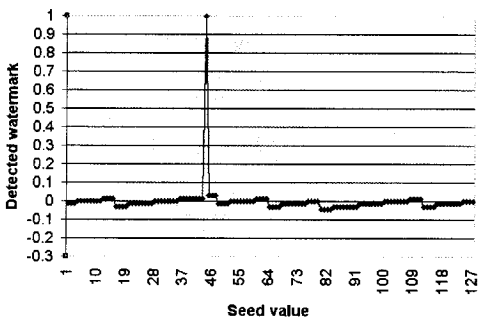


(b) Detected field labels for the original SMIL scene



(c) Located media carrying fields

(d) Original media content



(e) Detected watermark

Figure 5-5. Results from the original SMIL scene protection algorithm

Instead of shifting the whole scene by adding extra nodes and fields from the top and keeping the structure intact, two neutral extra nodes and four extra fields are added within the scene. The addition of those neutral nodes and fields is not supposed to affect the SMIL multimedia presentation, but by doing this, part of the scene structure has shifted in position from the rest of the scene structure. From Table 5-1, it can be seen that 46 % of the original structure labeling has changed and the labeling connection sequence has changed from that of the original as shown in Figure 5-7(a). The number of detected PRS bits was 66, but after error correction the remaining shifted fields were recognized and 90 PRS bits were recovered. The loss of information PRS bits occurred because of the relationship loss of some of the node with their parents nodes of the original scene due to part of the scene shift process All four media carrying fields were still detected after the inner structure shift, as revealed in Figure 5-7(b), but the media field labels FL3 and FL4, were less strongly detected due to their location in the shifted part of the structure, which affected its field labeling connection sequence with respect to the field labels on the non-shifted part of the structure. No other field has been mistakenly detected to be one of those four media carrying fields.

5.2.1.2. Scene Structure Cropping

The original scene was modified by removing a section at the end part of the SMIL scene structure. As it can be seen from Figure 5-8(a), the remaining section of the scene was still recognized as being part of the original one even though some of the nodes were deleted. From Figure 5-8(b), it can be seen that three media carrying fields were strongly detected even though a big section of the scene was deleted. But it can be observed that the detection signal of the three media carrying fields is not as strong as in the original scene, due to the removed part of the scene and the loss of some of the labeled fields located in the original scene structure. The fourth media carrying field was lost due to its location in the deleted part of the structure.

5.2.1.3. Scene Deformation

An extreme test was performed to check the robustness of the algorithm. The original scene structure was split into independent parts and interchanged. This is not suitable for someone tampering with the scene since this will destroy the significance of the SMIL multimedia presentation. By doing this structure deformation though, a significant number of the labeling connection sequence present in the original scene was lost. But as can be seen from Figure 5-9(a), some sections of the scene were still recognized as matching to parts of the original scene even though their positions were displaced – i.e. fields 10 to 16 matching with fields in the original scene but with higher field labeling value which signifies a shift forward in positions; and fields 19-24 and 28-31 matching with other parts of original scene but with lower labeling value, which signifies a backward shift in structure as expected. From Figure 5-9(b), it can be seen that the four media carrying fields were still detected even after that major structure deformation. It can also be seen that the media fields changed their position from that of the original. The field containing the Lena image "FL1" for instance, was located in the top part of the original scene structure and now it is the last to be detected (field label 48188) after being placed at the end of the scene structure.

5.2.1.4. A Totally Different Scene with(out) Original Scene

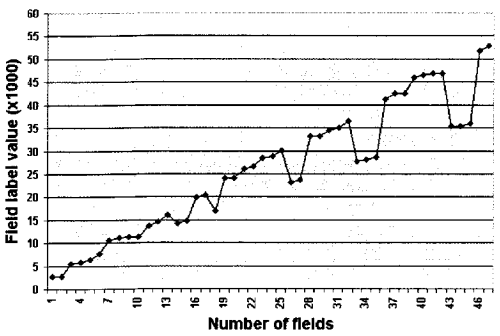
Given a totally different scene and being provided with the original signature, no similarity detection was made with the original scene, as shown in Table 5-1.

The original scene was then added to the end of that given scene structure, combining one scene with another one. From Table 5-1, it can be seen that most of the field labeling of the original scene was modified. But after applying the algorithm explained in section 5.1.1.3 subsection "Scene Structure Detection/Authentication Algorithm", 119 bits out of the total 121 original PRS bits were detected. From Figure 5-10(a), it can be seen that the field labeling sequence remained intact even though most of the labels were modified. From Figure 5-10(b), it is shown that out of the 403 fields located in the scene structure, the four specific media carrying fields were correctly detected, even though their field

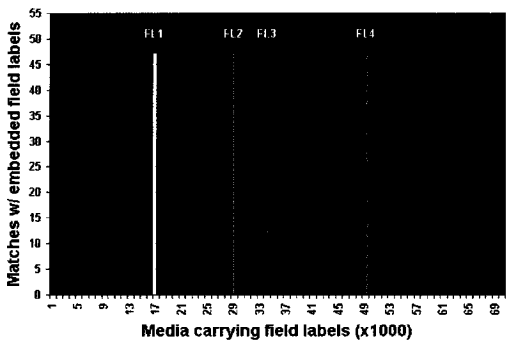
labels were modified and shifted due to the change of location with respect to the whole scene structure.

Structure modification	Nodes/Fields present after modification	Structure labeling modified	# of PRS bits recovered (no error correction)	# of PRS bits recovered (error correction)
Original (Figure 5-5)	89 Nds / 303 Flds	0 %	121 bits	121 bits
Top shifting (Figure 5-6)	91 Nds / 307 Flds	100 %	0 bits	120 bits
Inner shifting (Figure 5-7)	91 Nds / 307 Flds	46 %	66 bits	90 bits
Cropping (Figure 5-8)	63 Nds / 209 Flds	0 %	79 bits	79 bits
Deformation (Figure 5-9)	89 Nds / 303 Flds	94 %	9 bits	48 bits
Different scene	35 Nds / 108 Flds	N/A	0 bits	0 bits
Diff. Scene w/ orig. (Figure 5-10)	118 Nds / 403 Flds	96 %	8 bits	119 bits

Table 5-1. Effect of tampering attacks on a SMIL scene structure

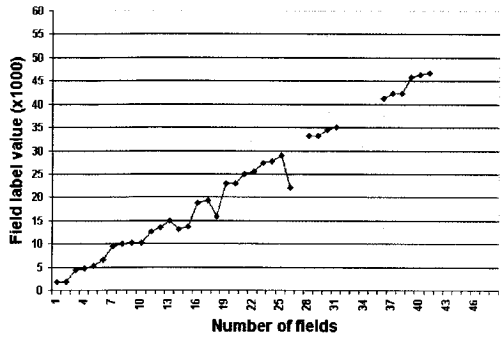


(a) Detected field labels for top shifted scene structure

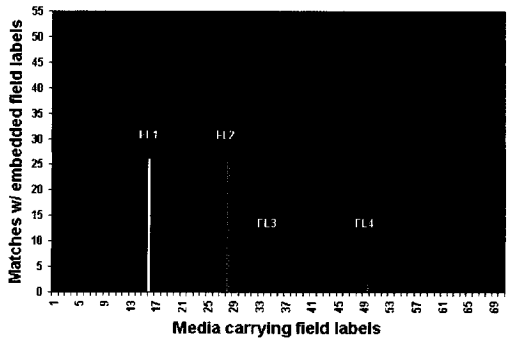


(b) Located media carrying fields

Figure 5-6. Detected/authenticated original scene and located media content after top shifting SMIL scene structure

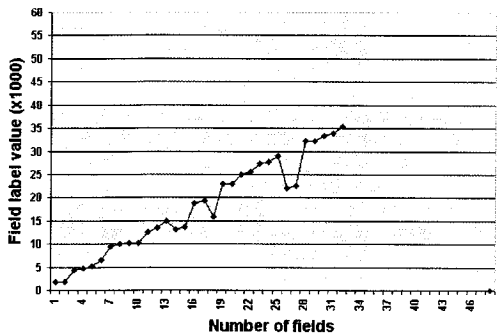


(a) Detected field labels for inner shifted scene structure

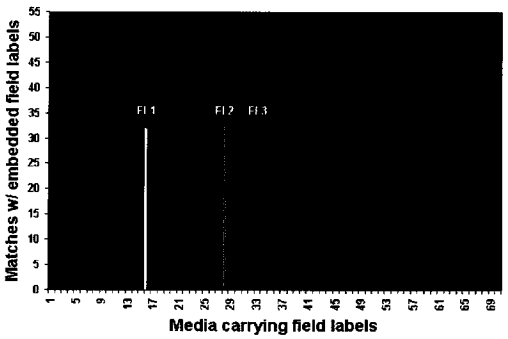


(b) Located media carrying fields

Figure 5-7. Detected/authenticated original scene and located media content after inner shifting SMIL scene structure

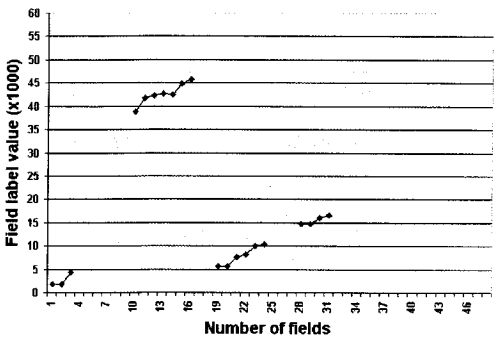


(a) Detected field labels for cropped scene

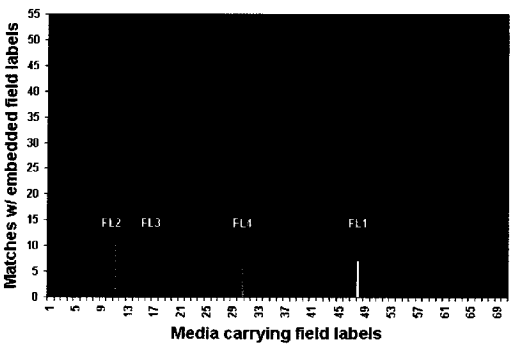


(b) Located media carrying fields

Figure 5-8. Detected original scene and located media content after scene cropping

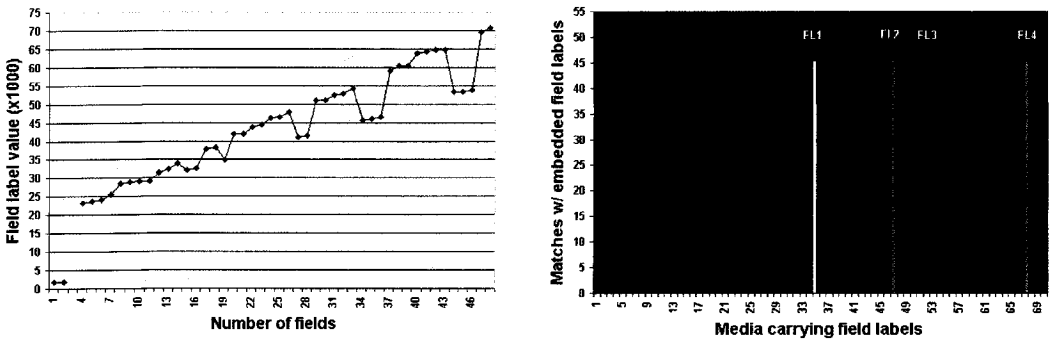


(a) Detected field labels for deformed scene



(b) Located media carrying fields

Figure 5-9. Detected/authenticated original scene and located media content after scene deformation



(a) Detected field labels of diff. w/ original scene combined

(b) Located media carrying fields

Figure 5-10. Detected/authenticated original scene and located media content when combined with a different scene

5.2.2. Test Case Two

This second test set is used to check for consistency and accuracy of the algorithm in detecting a given scene under scene structure modifications and show the uniqueness of the generated content signature for a given scene.

A content signature and located media carrying fields (with label values 14878, 24398, and 41558) were generated for a new SMIL scene structure, as shown in Figure 5-11. The PRS was generated using same polynomial $h(x) = x^7+x^5+x^4+x^3+1$, and with the same binary seed value = 0101100 used in test case one. The values of both α and β used for structure field labeling are both set to 1. For that given scene and the PRS, 115 PRS bits were mapped to 35 fields located throughout the SMIL scene structure. Different structural modifications were applied. As it can be seen from Table 5-2, and the results shown in the figures between Figure 5-12 and Figure 5-16, consistency of the output results to those shown in test case one is revealed.

After applying two nodes and four fields (attributes) to the top of the scene structure, the whole scene was shifted in position and the structure labeling sequence was totally

modified. From Table 5-2, it can be seen that 0 bits of the PRS were detected, but after applying the error correction /recovery process, 114 of the 115 bits were recovered. As it can be seen from the Figure 5-12, the SMIL scene structure was detected and the media carrying fields were located, even though their labeling values were changed (15958, 25478, and 42638) from that of the original scene.

A test was also made by applying two extra nodes and four extra fields to the middle of the scene structure without affecting the SMIL multimedia presentation. By doing this, the labeling sequence of the shifted part of the original scene was changed. But from Figure 5-13, it can be seen that the scene and its media carrying fields were still detected and authenticated.

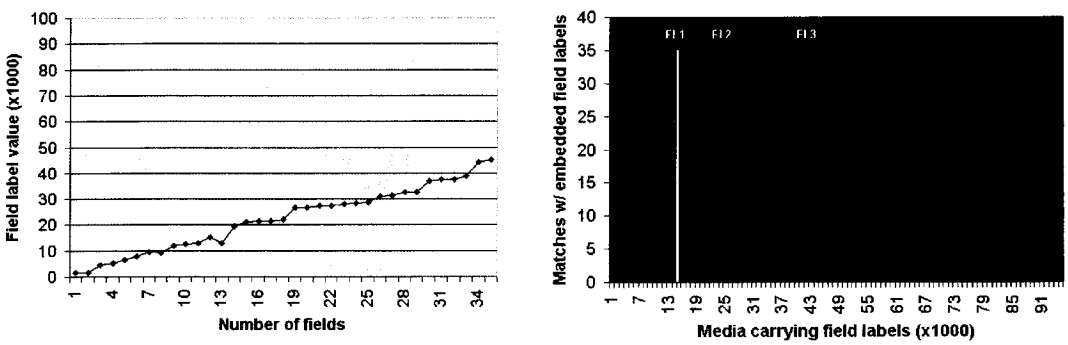
A top section of the original SMIL scene structure was then removed, and only a remaining section of the bottom part of the scene structure was left. From Figure 5-14, it can be seen that even though the remaining part of the scene structure and its structure labeling sequence were totally modified, that remaining part of the structure was still detected and authenticated. In Figure 5-14(b), it can be seen that only one media carrying field (FL3) was detected because the other fields were deleted with the cropped part of the scene.

The original scene structure was then split into two separate independent sections and interchanged. It should be noted that this is not suitable type of tampering attack, since it will ruin the playing order of the SMIL presentation. From Figure 5-15(a), it can be seen that the scene was still detected even though some parts of the scene structure were not located in the expected position as the original scene. The media carrying fields were also detected but their location was changed also due to the scene structure modification. From Figure 5-15(b), it can be seen that the media carrying field FL3 now lies before FL1 and FL2 due to the structure interchange.

A test was also made by combining the original scene structure used in test case one with the original scene structure used in test case two. The original scene used in test case one was added to the top of the scene structure. The content signatures, one for the scene used in test case one (shown in Figure 5-5(b)), and the other for the scene used in test case two (shown in Figure 5-11(a)), were both generated using the same polynomial $h(x) = x^7+x^5+x^4+x^3+1$ with the same seed value. When combining both scenes together, and using the content signature generated for the scene in the test case two, the algorithm was able to detect the scene part used in the test case two and not the other scene part used in test case one. This shows the uniqueness of the content signature generated for detecting a specific scene.

Structure modification	Nodes/Fields present after modification	Structure labeling modified	# of PRS bits recovered (no error correction)	# of PRS bits recovered (error correction)
Original (Figure 5-11)	76 Nds / 288 Flds	0 %	115 bits	115 bits
Top shifting (Figure 5-12)	78 Nds / 292 Flds	100 %	0 bits	114 bits
Inner shifting (Figure 5-13)	78 Nds / 292 Flds	45 %	64 bits	110 bits
Cropping (Figure 5-14)	43 Nds / 154 Flds	94 %	8 bits	54 bits
Deformation (Figure 5-15)	76 Nds / 288 Flds	94 %	8 bits	102 bits
Frst Org. Scene w/ Cur. orig. Scene (Figure 5-16)	156 Nds / 567 Flds	94 %	8 bits	106 bits

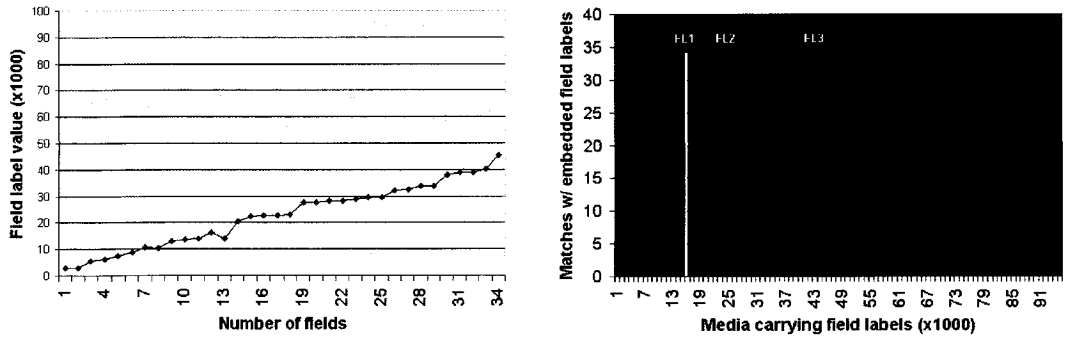
Table 5-2. Effect of tampering attacks on another SMIL scene structure



(a) Detected field labels of original scene structure

(b) Located media carrying fields

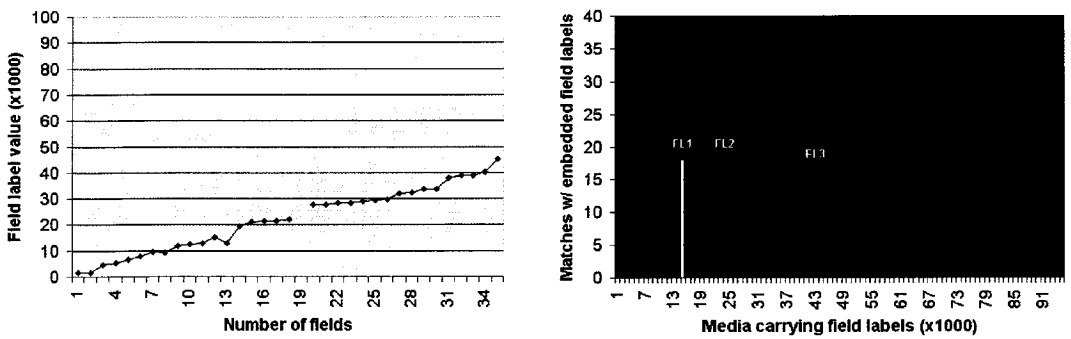
Figure 5-11. Detected/authenticated original scene and located media content



(a) Detected field labels for top shifted scene structure

(b) Located media carrying fields

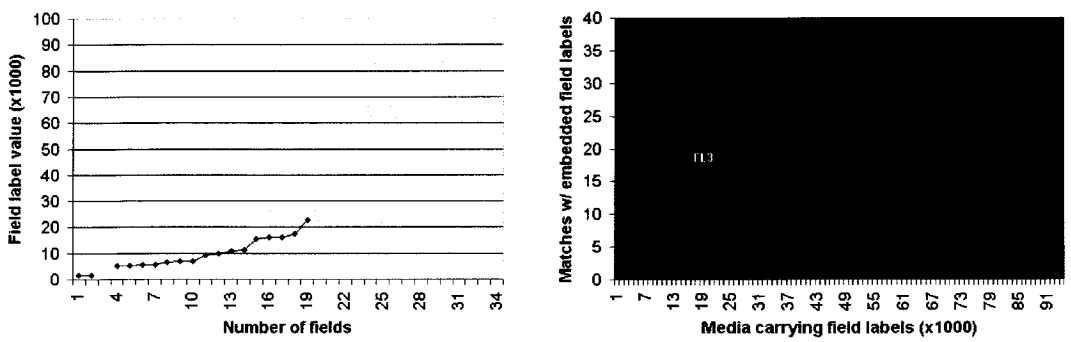
Figure 5-12. Detected/authenticated original scene and located media content after top shifting SMIL scene structure



(a) Detected field labels for inner shifted scene structure

(b) Located media carrying fields

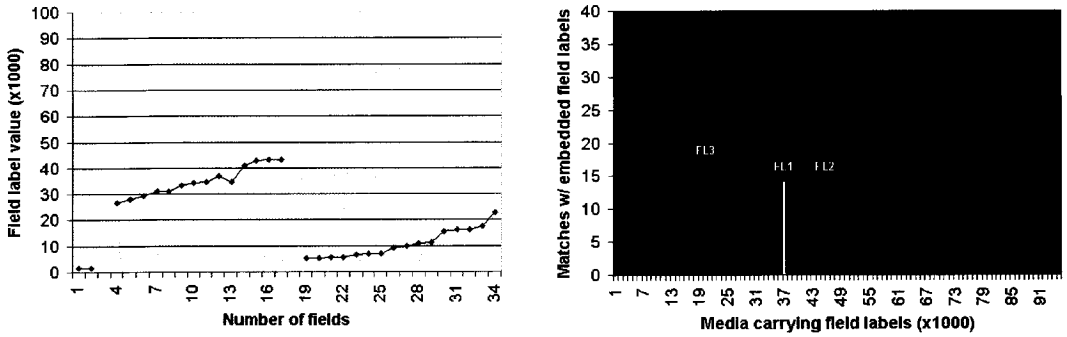
Figure 5-13. Detected/authenticated original scene and located media content after inner shifting SMIL scene structure



(a) Detected field labels for cropped scene

(b) Located media carrying fields

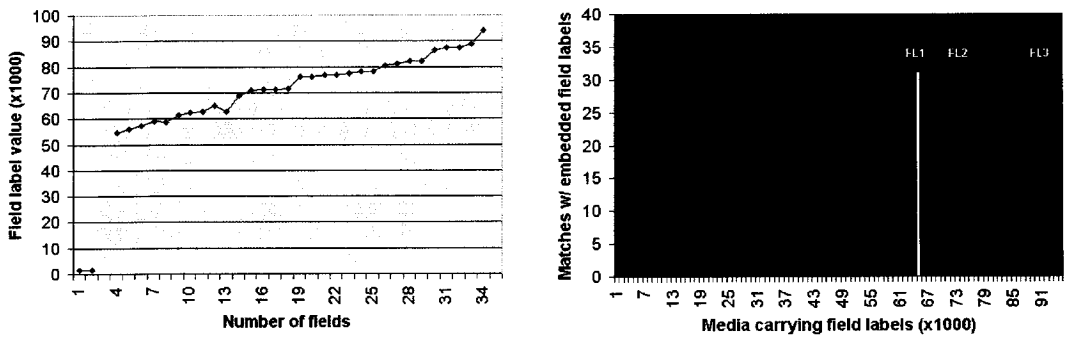
Figure 5-14. Detected original scene and located media content after scene cropping



(a) Detected field labels for deformed scene

(b) Located media carrying fields

Figure 5-15. Detected/authenticated original scene and located media content after scene deformation



(a) Detected field labels of diff. w/ original scene combined

(b) Located media carrying fields

Figure 5-16. Detected/authenticated original scene and located media content when combined with previous original scene in test case one

Chapter 6

MPEG-4 & SMIL Content Protection Algorithm

Chapter 4 and Chapter 5 explained in details the algorithms used for the authentication and protection of MPEG-4 and SMIL scene structures respectively. This chapter proposes an algorithm for the protection of the media content found within those scenes. Those media content can be pictures, texture mapping, photographs, etc. The proposed algorithm makes use of pseudo-random sequences and spread spectrum for media content protection.

Spread spectrum was first used by the military for communications, in which a narrow-band signal is spread over a much larger bandwidth. The exact form of the spreading is a secret known only by the transmitter and receivers. Without knowledge of the spreading function, it is almost impossible for an adversary to detect or interfere with a transmission. When spread spectrum is used for communications, the spreading function is designed to make the modulated message appear to an unauthorized user like background noise. This makes it very difficult for an unauthorized user to determine if a transmission is occurring.

Using the same method used in spread spectrum communications, spread spectrum can also be applied for watermarking media objects. Only the authorized user will be able to extract the watermark from a media object But for the unauthorized user, the watermark will appear as background noise in the media object. Even if tampering attack was made by adding extra noise to the media object, the added noise will not be enough to destroy the watermarking without damaging the object quality (i.e. image quality).

The following will provide detailed explanation of the proposed algorithm used for the copyright protection of the media content (such a pictures, image textures, photographs, etc.) used within the MPEG-4 and SMIL scenes.

6.1. Media Content Protection

An algorithm has been proposed for the protection of media content located within an MPEG-4 scene from unauthorized use and distribution. The algorithm makes use of pseudo random sequences and the spread spectrum technique for the protection of the media content. The algorithm is based on several methods obtained from a combination of papers [21] [37] [38] [114]. The same PRS used to protect the MPEG-4 XMT scene structure explained in chapter 4, is now used to watermark those media objects. The application of this algorithm is not only for copyright protection of the media content, but also to establish a relationship between the MPEG-4 scene and its media content when it comes to authentication and identification of the original owner. Even if the media object has been removed from its original scene and used in a different scene, it can still be identified to the actual scene it originally belonged to.

The algorithm can be extended, by applying a 2D PRS to the x and y coordinates of the media object (i.e. picture). By doing this, the algorithm will not only be able to copyright the object, but also locate the coordinates of any error or tampering attacks applied on the original copyrighted media object.

6.1.1. *Watermark Embedding*

Let the bit sequence a_j of the PRS be the watermark that will be embedded into the media objects (pictures, texture mapping, photographs, etc). Knowing the total number of pixels N in an image and using r as the chip-rate to spread the bit sequence a_j throughout the whole image, a total of N/r bits could be embedded. In order to generate a matching similarity between the PRS used for structure protection and the one used for media content protection, at least the bit sequence used to generate scene structure signature should be embedded into the media objects by adjusting the chip rate accordingly. After setting the chip rate factor r, the spread sequence b_i is obtained.

$b_i = a_j$ where $j.r \leq i \leq (j+1).r$

With a pseudo-noise sequence p_i randomly generated (used for frequency spreading) is then applied to the sequence b_i and then scaled by a factor α to obtain the spread spectrum watermark w_i:

$w_i = \alpha * b_i * p_i$

The image is converted to the frequency domain using the discrete cosine transform (DCT). The watermark w_i is then converted to the frequency domain using DCT and added to the image DCT coefficients V_i excluding the DC coefficients to obtain the watermarked image DCT coefficients:

$V'_i = V_i + W_i$

The inverse DCT transformation is then applied on V'_i to obtain the watermarked image.

6.1.2. Watermark Extracting

In order to extract the watermark, the original unwatermarked image (which is kept secret by the owner) is first subtracted from the watermarked one. The multiplication of the output image with a pseudo-noise signal p_i, used for watermark embedding, is then applied. This is followed by the correlation sum s_j, which will yield the jth information bit of the extracted PRS watermark.

$$s_j = \sum_{i=j.r}^{(j+1).r-1} p_i.w_i = \sum_{i=j.r}^{(j+1).r-1} p_i^2.\alpha.b_i = \alpha\,.\,r.\,a_j$$

$$\text{sign}(s_j) = \text{sign}(\alpha\,.\,r.\,a_j) = \text{sign}(a_j) = a_j$$

Since $\alpha > 0$, $r > 0$, $a_j = \pm 1$, then the correlation sum s_j will be positive if the jth embedded bit is +1, and negative if the jth embedded bit is –1.

6.2. Experimental Results

The same PRS used for scene authentication and media content location is also used for media content protection within the same scene. Even if the located media within the scene were modified, the algorithm should still be able to detect the embedded watermark, identify that media object and to which MPEG-4 scene it belonged. The PRS used in this experiment was generated using the polynomial $h(x) = x^7+x^5+x^4+x^3+1$ with a seed value of 0101100. It was embedded into the Lena portrait of size 256x256 pixels located within the MPEG-4 scene shown in Figure 4-11(a). The media content protection algorithm, explained in section 6.1, was used for embedding the PRS watermark with a chip-rate r set to 500 and scaling factor α equal to 1. The chip-rate was set accordingly in order for the entire PRS, used for structure authentication, to be embedded in the media object.

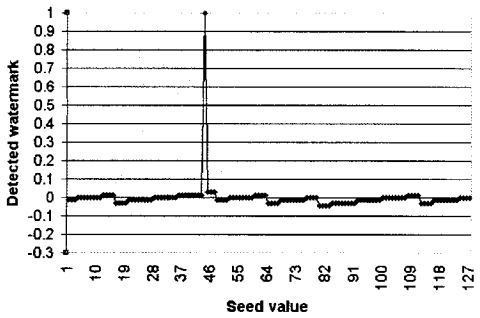
From Figure 6-1, it can be seen that even after applying JPEG compression to the portrait, the watermark was still detected. The PRS watermark was tested against other pseudo-random sequences using the same generating polynomial $h(x) = x^7+x^5+x^4+x^3+1$ but with a different seed. As it can be seen from Figure 6-1(b), only by knowing the generating polynomial and the exact seed value used (binary 0101100 <==> decimal 44) that the watermark can be extracted.

Since the watermark was added to all the image DCT coefficients excluding the DC ones, it can be seen that the watermark was strongly detected even after high pass filtering of the portrait as displayed in Figure 6-2. But when the portrait has gone through low pass filtering as shown in Figure 6-3, the watermark can still be detected but not as strong since most of the high frequency coefficients were removed.

The content protection algorithm was also tested against cropping. As it can be seen from Figure 6-4(a), 128x128 pixels were removed from the image. Even though such a considerable size was removed from the portrait, the watermark was still strongly detected considering the amount of lost information (including part of the embedded watermark) in the cropped section.

The algorithm was also tested by inserting the same PRS generated from the polynomial $h(x) = x^7+x^5+x^4+x^3+1$ using 126 different seed values displayed in Figure 6-5. As it can be seen, the original embedded watermark was still strongly detected, shown in Figure 6-5(b), and only the true owner of the original portrait that contains no watermarks can prove ownership of the object in case of dispute.

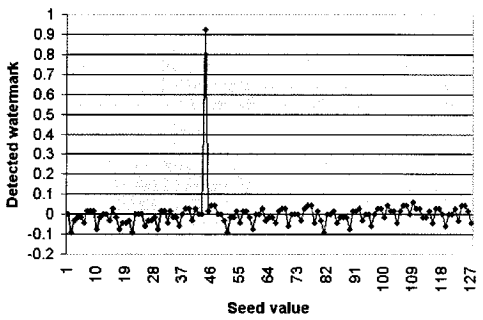
(a) Portrait after JPEG compression



(b) Detected watermark after JPEG compression

Figure 6-1. Portrait and detected watermark after applying JPEG compression

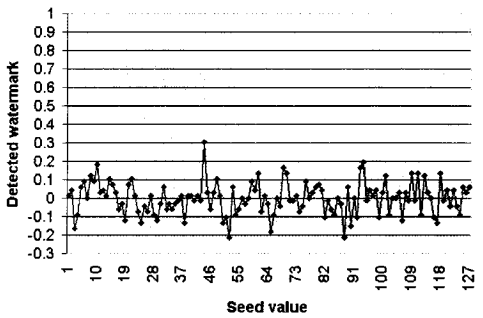
(a) Portrait after high-pass filtering



(b) Detected watermark after high-pass filtering

Figure 6-2. Portrait and detected watermark after applying high-pass filtering

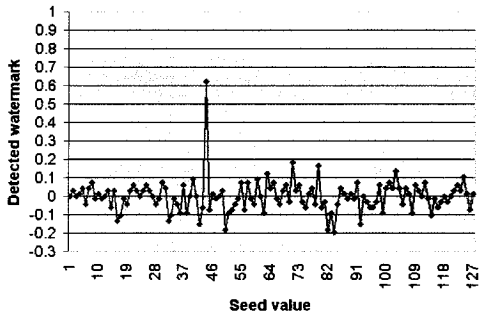
(a) Portrait after low-pass filtering



(b) Detected watermark after low-pass filtering

Figure 6-3. Portrait and detected watermark after applying low-pass filtering

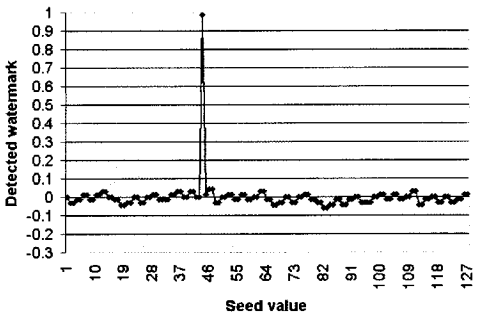
(a) Portrait after cropping



(b) Detected watermark after cropping

Figure 6-4. Portrait and detected watermark after cropping

(a) Portrait after inserting multiple watermarks



(b) Detected watermark after inserting multiple watermarks

Figure 6-5. Portrait and detected watermark after inserting multiple watermarks

Chapter 7

Conclusion and Future Extensions

This chapter provides the conclusion and summarizes the achievements in this thesis research work. In addition, the most important of possible future extensions are also discussed.

7.1. Conclusion

The development of the MPEG-4 standard has received a lot of attention for the development and processing of multimedia applications in recent years. Unlike the older standards such as MPEG-1 and MPEG-2, MPEG-4 is not a frame-based but an object-based framework. It allows for the coding of audio-visual objects. These objects can be audio, video, 3D models, or other multimedia data. The complete scene depends on the scene graph, which contains all the multimedia data and the detailed information for each object for the scene to be displayed properly.

In this thesis, an MPEG-4 player was developed that is able to open a scene whether it is in the MPEG-4 BIFS format or the MPEG-4 XMT format. The scene can also be updated/modified with nodes and/or parts of the scene added, deleted, or translated in the scene with all modification viewed on the display scene graph. The MPEG-4 scene can later be saved in any of the desired formats whether the MPEG-4 BIFS for enhanced compression or the MPEG-4 XMT for ease of editing and modification.

Furthermore, in this thesis research work, a new robust algorithm for the authentication and protection of MPEG-4 scenes and their content has also been designed and implemented. Through a common secret generated PRS, the algorithm is able to authenticate the MPEG-4 scene, locate any watermarked media content, and copyright/protect those media content located with the MPEG-4 scene.

The algorithm is applied on the MPEG-4 XMT-A scenes. Even though the XMT structure lacks the noise components, which are considered a vital part for information hiding in media objects such as image, audio and video scenes, the proposed algorithm still proved to be robust against many forms of scene structure modifications and tampering attacks. The algorithm made use of pseudo-random encoding sequences and MPEG-4 BIFS encoding standard to generate two unique content-based signatures for a given scene. One generated signature was used to authenticate a scene later on whenever required, and the other signature was used to locate all the media carrying fields located within that protected scene. The experimental results revealed the robustness of the algorithm in detecting a given scene and locating any media carrying fields even after applying different structural modifications. The experiments also showed the uniqueness of the generated signatures for a given scene. It was shown that even if two authentication signatures for two different scenes were generated from the same PRS using the same primitive polynomial and with the same seed value for the PRS, if both of those scenes were combined together later on in one scene, using the appropriate authentication signature, the algorithm was able to detect and authenticate that part of the scene corresponding to that signature from the rest of the scene.

Moreover, the idea of watermarking images using the PRS has been proposed previously by Tirkel et al. [114], but the method of having a common secret PRS code that is able to authenticate the scene, find the media content and protect it, and establish a relationship between the MPEG-4 scene and its media content is totally new. Even if a media object was removed from the MPEG-4 scene and used in a different scene, the algorithm can

still extract the watermark from that object and relate it to the original scene it was first used in.

The MPEG-4 scene authentication and content protection algorithm was also generalized to authenticate and protect SMIL scenes and their content. SMIL is an XML-based language used to write interactive multimedia presentations. Unlike MPEG-4 XMT scene structure where a field (MFfield or SFfield) can contain other nodes in the hierarchical ordered tree, in SMIL only nodes are linked to other nodes and fields are related to their parent's node and not to any other nodes in the structure. This has to be taken into consideration when designing the SMIL scene protection algorithm.

The SMIL protection algorithm can be applied on any XML-based language with predefined nodes and fields/attributes.

To the best of our knowledge, the algorithms designed for the protection of MPEG-4 and SMIL scenes, are the first up to now that proved to be robust against different tampering attacks while still being able to authenticate given scenes and protect their media content.

7.2. Summary of Contributions

The following list summarizes the contributions of this thesis:

- Design and implementation of a MPEG-4 XMT parser
- Design and implementation of a MPEG-4 BIFS ⇔ XMT encoder/decoder tool.
- Design and implementation of a MPEG-4 player with real-time scene structure update to any modifications applied to the scene.
- Design and implementation of a MPEG-4 scene authentication and media content location and protection algorithm.
- Design and implementation a SMIL scene authentication and media content location protection algorithm.

- Implementation of a watermarking algorithm for the protection of still images (pictures, textures, etc) located in the MPEG-4 or SMIL scenes.

7.3. Future Research

The following are possible future extensions/enhancements to this thesis research work:

- The MPEG-4 and SMIL scene authentication schemes used are symmetric schemes and cannot be authenticated by the public. This work can be enhanced by generating public verifiable signatures for scene authentication.
- The content-based authentication algorithm applied on SMIL scenes can be enhanced to support any XML-based scene.
- A more robust algorithm for watermarking images can be implemented.
- A common watermarking algorithm can be applied to images and other media content (i.e. video, audio, 3D models, etc.) located within an MPEG-4 scene.

References

[1] M. Arnold and S. Kanka, "MP3 robust audio watermarking," *in DFG VIIDII Watermarking Workshop*, 1999.

[2] L. Babai, E. Luks, "Canonical labeling of graphs," *in Fifteens Annual ACM Symp. on Theory of Comp.*, pp. 171-183, 1983.

[3] R. Balakrishnan, A. Selvam, V. Yegnanarayanan, "On felicitous labelings of graphs," *in Proc. National Workshop on Graph Theory and its Appl.*, pp.47-61, 1996.

[4] J. R. Ball, A. H. Spittle, H. T. Liu, "High-Speed m-Sequence Generation: a Further Note," *in Electron. Lett.*, vol. 11, pp. 107-108, 1975.

[5] M. Barni, F. Bartonlini, V. Cappellini, and N. Checcacci, "Object watermarking for MPEG-4 video streams copyright protection," *in Proc. of SPIE*, vol. 3671, January, 2000.

[6] P. Bas, J.M. Chassery, and F. Davoine, "Using the fractal code to watermark images," *in Proc. of ICIP*, pp. 469-473, 1998.

[7] J. T. B. Beard, "Computing in GF(q)," *Math. Comput.*, vol. 28, pp. 1159-1166, 1974.

[8] W. Bender, D. Gruhl, and N. Morimoto, "Techniques for data hiding," *in Proc SPIE,* vol. 2420, pp. 40, Feb. 1995.

[9] O. Benedens, "Geometry-based watermarking of 3D models," *in IEEE Computer Graphics and Applications*, pp. 46-55, January 1999.

[10] E. R. Berlekamp, "Algebraic Coding Theory". *New York: McGrawHill*, 1968.

[11] D. Bohen and J. Shaw, "Collusion-secure fingerprinting for digital data," *in Proceedings of CRYPTO*, pp. 452-465, 1995.

[12] D. Bohen and J. Shaw, "Collusion-secure fingerprinting for digital data," *in IEEE Trans. on Information Theory*, vol. 44, no. 5, pp. 1897-1905, 1998.

[13] L. Boney, A.H. Tewfik, and K.N. Hamdy, "Digital watermarks for audio signals," *in Int. Conf. on Multimedia Computing and Systems*, 1996.

[14] A. Bors and I. Pitas, "Image watermarking using DCT domain constraints," *in Proc. Int. Conf. Image Processing (ICIP)*, Sept. 1996.

[15] J. Brassil, S. Low, N. Maxemchuk, L. O'Gorman, "Electronic marking and identification techniques to discourage document copying," *in Proc. of IEEE INFOCOM'94*, vol. 3, pp. 1278-1287, June, 1994.

[16] D. C. Buchthal, D. E. Cameron, "Modern Abstract Algebra", *PWS Publishers*. 1987.

[17] C. Busch, W. Funk, and S. Wolthusen, "Digital watermarking form concepts to real-time video applications," *in IEEE Comput. Graphics. Applicat.*, pp.25-35, January 1999.

[18] D. E. Carter, "On The Generation of Pseudo-Noise Codes," *in IEEE Trans. Aerosp. Electron. Syst.*, vol. 10, pp. 898-899, 1974.

[19] P. Cheeseman, B. Kanefsky and W.M. Taylor, "Where the really hard problem are," *in Proc. of the International Joint Conf. on Artificial Intelligence*, vol. 1, pp. 331-337, 1991.

[20] C. Collberg and C. Thomborson, "On the limits of software watermarking," Technical Report #164, Department of Computer Science, The University of Auckland, Auckland, New Zealand, August 1998.

[21]I. J. Cox, J. Kilian, F.T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *in IEEE Trans. Image Processing*, vol. 6, pp. 1673-1687, 1997.

[22] S. Craver, N. Memon, B. L. Yeo and M. Yeung, "Can invisible watermarks resolve rightful ownerships ?" *Technical Report RC 20509, IBM Research Division*, July 1996.

[23] I. G. Cumming, "Autocorrelation Function and Spectrum of a Filtered, Pseudo-Random Binary Sequence," *in Proc. Inst. Elect. Eng.*, vol. 114, pp. 1360-1362, 1967.

[24] V. Darlagiannis and N. D. Georganas, "Virtual Collaboration and Media Sharing using COSMOS," *in Proc. 4th World Multiconference on Circuits, Systems, Communications and Computers (CSCC 2000)*, Greece, July 2000.

[25] R. L. Davidson and N. Myhrvold, "Method and system for generating and auditing a signature for a computer program," US Patent 5,559,884, September 1996. Assignee: Microsoft Corporation.

[26] I. Defee, J. Kangasoja, M. Rustari, "COMIQS System for commercial presentations on the Internet", ECMAST, 1999.

[27] F. Deguillaume, G. Csurka, J. O'Ruanaidh and T. Pun, "Robust 3D DFT video watermarking," *in IS&T/SPIE's 11th Annual Sumposium, Electronic Imaging: Security and Watermarking of Multimedia Contents*, vol. 3657, pp. 23-29, January 1999.

[28] T. Deretsky, S. M. Lee, J. Mitchem, "On vertex prime labelings of graphs," *in Graph Theory Combinatorics and Applications*, Wiley, NewYork, vol. 1, pp. 359-369, 1991.

[29] J. Dittmann, M. Stabenau, and R. Steinmetz, "Robust MPEG video watermarking technologies," *in Proc. ACM Multimedia*, September 1998.

[30] J. Dittmann, M. Steinebach, and R. Steinmetz, "Digital watermarking for MPEG Audio Layer 2," *in Workshop of Multimedia and Security at ACM Multimedia '99*, pp. 117-122, 1999.

[31] S. Fredricsson, "Pseudo-Randomness Properties of Binary Shift Register Sequences," *in IEEE Trans. Inform. Theory*, vol. 21, pp. 115-120, 1975.

[32] J. Fridrich, "Methods for data hiding," *Technical report*, Center for Intelligent Systems, SUNY Binghamton, USA, 1997.

[33] S. W. Golomb, "How to number a graph," *in Graph Theory and Computing, Academic Press*, New York, pp. 23-37, 1972.

[34] D.H. Green, I.S. Taylor, "Irreducible Polynomials over Composite Galois Fields and their Application in Coding Techniques", *in Proc. Inst. Elec. Eng.*, vol. 121, pp. 935-939, 1974.

[35] H.Guo and N.D.Georganas, "Digital image watermarking for joint ownership", *in Proc. ACM Multimedia 2002*, December 2002.

[36] F. Hartung, P. Eisert, B. Girod, "Digital watermarking of MPEG-4 facial animation parameters," *in Computers and Graphics – Special Issue on Data Security in Image Communications*, vol. 22, no. 4, pp. 425-435, 1998.

[37] F. Hartung and B. Girod, "Digital watermarking of MPEG-2 coded video in the bitstream domain," *in Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, vol. 4, pp. 2621-2624, April 1997.

[38] F. Hartung and B. Girod, "Digital watermarking of raw and compressed video," *in Proc. of the European EOS/SPIE Symposium on Advanced Imaging and Network Technologies*, October, 1996.

[39] F. Hartung and B. Girod, "Fast public-key watermarking of compressed video," *in Proc. IEEE Int. Conf. on Image Processing*, vol.1, pp. 528-531, October 1997.

[40] J. T. Harvey, "High-Speed m-Sequence Generation," *Electron. Lett.*, vol. 10, pp. 480-481, 1974.

[41] K. Holmes, "Computer software protection," US Patent 5,287,407, February 1994. Assignee: International Business Machines.

[42] H. Hoppe, "Progressive Meshes," *in ACM SIGGRAPH 96 Conf. proceedings*, pp. 99-108, August 1996.

[43] M. Hosseini and N. D. Georganas, "Suitability of MPEG4's BIFS for Development of Collaborative Virtual Environments," *in IEEE International Workshop on Enabling Technologies for Collaborative Enterprises (WET ICE'01)*, June 2001, Cambridge, MA,USA.

[44] S. Inoue, K. Makino, I. Murase, O. Takizawa, T. Matsumoto, and H. Nakagawa, " A proposal on information hiding method using XML," *in the 1st Workshop on NLP and XML*, November, 2001.

[45] S. Inoue, K. Makino, I. Murase, O. Takizawa, T. Matsumoto, and H. Nakagawa, "A proposal on steganography methods using XML," *in The 2002 Symposium on Cryptography and Information Security*, pp.301-306, January 2002.

[46] S. Inoue, K. Makino, I. Murase, O. Takizawa, T. Matsumoto, and H. Nakagawa, "Proposal on information hiding method using XML", *in The 1st Workshop on NLP and XML*, November 2001.

[47] ISO/IEC 13818-1, "Information Technology – Generic coding of moving pictures and associate audio: Systems", ISO/IEC JTC1/SC29/WG11 N0801rev, April 1995.

[48] ISO/IEC 14496-1, "Information Technology – Coding of audio-visual objects, Part-1: Systems", 2001.

[49] ISO/IEC 14496-2, "Information Technology – Coding of audio-visual objects, Part-2: Visual", 2001.

[50] ISO/IEC 14496-3, "Information Technology – Coding of audio-visual objects, Part-3: Audio", 2001.

[51] ISO/IEC 14496-4, "Information Technology – Coding of audio-visual objects, Part-4: Conformance Testing", 2000.

[52] ISO/IEC 14496-5, "Information Technology – Coding of audio-visual objects, Part-5: Reference", 2001.

[53] ISO/IEC 14496-6, "Information Technology – Coding of audio-visual objects, Part-6: Delivery Multimedia Integration Framework (DMIF)", 2000.

[54] M. I. Jinnah, G. S. Singh, "A note on arithmetic numberings of graphs," *in Proc. Symposium on Graphs and Combinatorics*, pp. 83-87, 1991.

[55] S. Katzenbeisser and F.A.P. Petitcolas, "Information hiding techniques for steganography and digital watermarking," *Artech House*, January 2000.

[56] M. Kim, S. Wood and L. Cheok, "Extensible MPEG-4 Textual Format (XMT)," *in ACM Multimedia*, 2000.

[57] L. Kobbelt, S. Campagna, and H.P. Seidel, "A General Framework for Mesh Decimation," *in Proceedings of Graphics Interface*, 1998.

[58] E. Koch and J. Zhao, "Towards robust and hidden image copyright labeling," *in Proc. IEEE Workshop on Nonlinear Signal and Image Processing*, pp. 452-455, June 20-22, 1995.

[59] D. K. Koukopoulos and Y.C. Stamatiou, "A compressed-domain watermarking algorithm for MPEG audio layer 3," *in Proc. of ACM Multimedia Workshops*, pp. 7-10, October 5, 2001.

[60] D. Kundur and D. Hatzinakos, "A robust digital image watermarking method using wavelet-based fusion," *in Proc. of the Int. Conf. on Image Proceeding*, vol. 1, pp.544-547, Oct. 1997.

[61] C. Kurah, and J. McHughes, "A cautionary note on image downgrading," *in IEEE Computer Security Applications Conference 1992, Proceedings*, vol. 2, pp.153-159, 1992.

[62] M. Kutter, F. Jordan, and F. Bossen, "Digital signature of color images using aplitude modulation," *in Proc. of SPIE storage and retrieval for image and video databases*, no. 3022-5, pp. 518-526, February 13-14, 1997.

[63] A. Lang, S. Thiemert, E. Hauer, H. Liu, and F. Petitcolas, "Authentication of MPEG-4 data: risks and solutions," *in Proceedings of SPIE*, vol. 5020, 452-461, 2003.

[64] G. C. Langelaar, R. L. Lagendijk and J. Biemond, "Real-time labeling of MPEG-2 compressed video," *in Journal of Visual Communication and Image Representation*, August 1998.

[65] S. Lin, D. J. Costello, "Error Control Coding, Fundamentals and Applications", *Prentice-Hall, Inc.*, Englewood Cliffs, N. J. 1983.

[66] S. Low, N. Maxemchuk, "Performance comparison of two text watermarking methods" *in IEEE Jour. Select. Areas Commun.*, vol.16, pp.561-572, May 1998.

[67] S. Low, N. Maxemchuk, J. Brassil, L. O'Gorman, "Document marking and identification using both line and word shifting," *in Proc. of IEEE INFOCOM'95*, Boston MA, April, 1995.

[68] S. Low, N. Maxemchuk, A. Lapone, "Document identification for copyright protection using centroid detection," *in IEEE Trans. on Commun.*, vol.46, no.3, pp.372-383, 1998.

[69] C. Lu, S. Huang, C Sze and H. Liao, "Cocktail watermarking for digital image protection," *in IEEE Trans. on Multimedia*, vol. 2, no. 4, pp.209-224, 2000.

[70] J. Lucy, N. Rump, P. Kudumakis, "MPEG-4 Intellectual Property Management and Protection (IPMP) Overview and Applications," MPEG doc. ISO/IEC JTC1/SC29/WG11/N2614, Dec. 1998.

[71] X. Ma, "A graceful numbering of a class of graphs," *in J. Math. Res. and Exposition*, pp. 215-216, 1988.

[72] F. J. MacWilliams, N. J. A. Sloane, "Pseudo-Random Sequences and Arrays", *Proc. IEEE.* vol. 64, no. 12. Dec. 1976.

[73] F. J. MacWilliams, N. J. A. Sloane, "The theory of error-correcting codes," *Elsevier Science Publishers B.V.*, 1977.

[74] N. Maxemchuk, and S. Low, "Marking text documents," *in Proc. IEEE Int. Conf. Image. Processing 97*, vol. 3, pp. 13-16, Oct. 1997.

[75] R. J. McEliece, "Finite fields for computer scientists and engineers", *Kluwer Academic Publishers*, 1987.

[76] N. Memon and P.W. Wong, "Protecting digital media content," *in Communications of the ACM*, vol. 41, pp. 35-43, July 1998.

[77] M. Miller, I. J. Cox, and J. Bloom, "Watermarking in the real world: An application to DVD," *in Proc. WKsp. Multimedia and Security at ACM Multimedia*, Sept. 1998.

[78] K. Mitsui and K. Tanaka, "Video-steganography: How to secretly embed a signature in a picture," *in IMA Intellectual Property Project Proceedings*, vol.1, no.1, pp.187-205, 1994.

[79] S. Moller, A. Pfitzmann, and I. Stirand, "Computer based steganography: How it works and why therefore any restrictions on cryptography are nonsense, at best," *in Information Hiding: First International Workshop, Proceedings*, vol. 1174, pp. 7-21, 1996.

[80] S. A. Moskowitz and M. Cooperman, "Method for stegacipher protection of computer code," US Patent 5,745,569, January 1996. Assignee: The Dice Company.

[81] MPEG-7. "MPEG-7: Context, Objectives and Technical Roadmap, V.12," ISO/IEC JTC1/SC29/WG11 N2861, July1999.

[82] D. Nicholson, P. Kudumakis, and J.F. Delaigle, "Watermarking in the MPEG-4 context", *in Proc. Lecture Notes in Computer Science*, pp. 472-492, May, 1999.

[83] R. Ohbuchi, H. Masuda, and M. Aono, "Watermarking three-dimensional polygonal models," *in Proc. of the ACM MultiMedia*, pp. 261-272, 1997.

[84] R. Ohbuchi, H. Masuda, and M. Aono, "Watermarking three-dimensional polygonal models through geometric and topological modifications," *in IEEE Journal on Selected Areas in Communications*, pp. 551-560, May 1998.

[85] R. Ohbuchi, S. Takahashi, T. Miyazawa, and A. Mukaiyama, "Watermarking 3D polygonal meshes in the mesh spectral domain," *in Proc. of the Graphics Interface*, pp. 9-17, June 2001.

[86] J. Palsberg, S. Krishnaswamy, M. Kwon, D. Ma, Q. Shao, and Y. Zhang, "Experience with software watermarking," *in Proc. of 16th Annual Computer Security Applications Conference*, pp. 308-316, 2000.

[87] F.A.P. Petitcolas, Steganographic software,

http://www.cl.cam.ac.uk/~fapp2/steganography/stego_soft.html

[88] E. M. Petriu, J. S. Basran, F. C. A. Groen, "Automated Guided Vehicle Position Recovery", *in IEEE Trans. Instr. Measur.*, vol 39, no. 1, Feb. 1990.

[89] E. Petriu, *Z. Sakr*, H.J.W. Spoelder, and A. Moica, "Object recognition using pseudo-random color encoded structured light," *in Proc. IMTC/2000, IEEE Instrum. Meas. Technol. Conf.*, pp. 1237-1241, May 2000.

[90] I. Pitas and T. Kaskalis, "Applying signatures on digital images," *in Proc. IEEE Workshop Nonlinear Image ans Signal Processing*, pp. 460-463, June 1995.

[91] A. Piva, R. Caldelli, and A.D. Rosa, "A DWT-based object watermarking system for MPEG-4 video streams," *in Proc. Of ICIP'2000*, vol. 3, pp.5-8, September, 2000.

[92] C.I. Podilchuk and W. Zeng. "Perceptual watermarking of still images," *in Proc. Electronic Imaging*, vol. 3016, San Jose, CA, USA, February 1996.

[93] C.I. Podilchuk and W. Zeng, "Digital image watermarking using visual models," *in Proc. of the SPIE Conf. on Human Vision and Electronic Imaging II*, vol. 3016, pp. 100-111, Feb. 10-13, 1997.

[94] C.I. Podilchuk and W. Zeng. "Image-adaptive watermarking using visual models," *in IEEE Journal on Selected Areas in Communications*, vol.10, pp. 525--539, May 1998.

[95] J. Popovic, and H. Hoppe, " Progressive Simplicial Complexes," *in ACM SIGGRAPH 97 Conference Proceedings*, pp. 217-224, August 1997.

[96] E. Praun, H. Hoppe, and A. Finkelstein, "Robust mesh watermarking," *in SIGGRAPH 99 Proceedings*, pp.69-76, 1999.

[97] J. Puate and F. Jordan, "Using fractal compression scheme to embed a digital signature into an image," *in Proc. of SPIE Photonics East'96 Symposium*, 1996.

[98] L. Qiao and K. Nahrstedt, "Non-invertible watermarking methods for MPEG encoded audio," *in Technical Report No. UIUCDCS-R-98-2069*, Department of Computer Science, University of Illinois at Urbana-Champaign, June 1998.

[99] P. D. Roberts, R. H. Davis, "Statistical Properties of Smoothed Maximal-Length Linear Binary Sequences," *in Proc. Inst. Elec. Eng.*, vol. 113, pp. 190-196, 1966.

[100] S. Roman, "Coding and information theory," Springer-Verlag, 1992.

[101] J. Ruanaidh and T. Pun, "Rotation, scale and translation invariant digital image watermarking," *in Proc. IEEE Int. Conf. on Image Processing 1997 (ICIP 97)*, vol. 1, pp. 536-539, October 1997.

[102] R.J. Safranek, "Perceptually based prequantization for image compression," *in Proc. of SPIE Conference on Human Vision, Visual Processing and Digital Display V*, 1994.

[103] P.R. Samson, "Apparatus and method for serializing and validating copies of computer software," US Patent 5,287,408, Assignee: Autodesk, Inc., February 1994.

[104] Z. Sakr and N.D. Georganas, "A MPEG-4 XMT scene structure algorithm for authentication and copyright protection," *in IEEE 14th International Workshop on Research Issues on Data Engineering (RIDE WS-ECEG'2004)* , 2004, pp. 48-55.

[105] Z. Sakr and N.D. Georganas, "A MPEG-4 XMT algorithm for scene structure authentication and content location," *in Proceedings of the Canadian Conference on Electrical and Computer Engineering*, 2004, May.

[106] R. Sion, M. Atallah, S. Prabhakar, "On watermarking semi-structures," in Technical Report – CERIAS TR 2001-54.

[107] SMIL 1.0. "Synchronized Multimedia Integration Language (SMIL) 1.0 Specificatioin," *W3C Recommendation*, http://www.w3.org/TR/REC-smil/, June 1998.

[108] SMIL 2.0. "Synchronized Multimedia Integration Language (SMIL) 2.0 Specificatioin." *W3C Recommendation*, http://www.w3.org/TR/smil20/, August 2001.

[109] V. Solachidis, A. Tefas, N. Nikolaidis, S. Tsekeridou, A. Nikolaidis, I.Pitas, ``A benchmarking protocol for watermarking methods,'' *in Int. Conf. on Image Processing (ICIP'01)*, pp. 1023-1026, 7-10 October, 2001.

[110] M.D. Swanson, B. Zhu, and A.H. Tewfik, "Multiresolution scene-based video watermarking using perceptual models," *in IEEE Journal On Selected Areas in Communications*, vol.16, no. 4, May 1998.

[111] B. Tao and B. Dickinson, "Adaptive watermarking in the DCT domain," *in Proc. Int. Conf. Image Processing (ICIP)*, Sept. 1996.

[112] A. Tefas and I. Pitas, "Robust image watermarking using progressive detection," *in Proc. of 2001 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 7-11 May 2001.

[113] F. Tip, C. Laffra, P.F. Sweeney, and D. Streeter, "Practical experience with an application extractor for Java," *IBM Systems Journal*, 1999.

[114] A.Z. Tirkel and C.F. Osborne, "Image watermarking – a spread spectrum application," *in Proc. IEEE 4th Int. Symp. on Spread Spectrum Techniques and Applications*, vol.2, pp. 785-789, 1996.

[115] A. Tirkel, G. Rankin, R. van Schyndel, W. Ho, N. Mee and C. Osborne, "Electronic water mark," *in Proc. DICTA*, pp. 666-672, Dec. 1993.

[116] A. Tirkel, R. van Schyndel, and C. Osborne, "A two-dimensional watermarking," *in Proc. DICTA*, 1993.

[117] N. Trif, "Model-Based Visual Recognition of 3D Objects using Pseudo-Random Grid Coding", Masters thesis, University of Ottawa, May 1993.

[118] R. Van Schyndel, A. Tirkel and C. Osborne, "A digital watermark," *in Proc. Int. Conf. ImageProcessing (ICIP)*, vol.2, pp. 86-89, 1994.

[119] G. Voyatzis and I. Pitas, "Applications of toral automorphisms in image watermarking," *in Proc. Int. Conf. Image Processing*, vol. 3, pp. 237-240, Sept. 1996.

[120] M.G. Wagner, "Robust Watermarking of Polygonal Meshes," *in Proceedings of the Geometric Modeling and Processing 2000*, April 2000.

[121] A. E. Walsh, M. Bourges-Sevenier, "MPEG-4 Jump Start," *Prentice Hall Inc.*, 2002.

[122] A.B. Watson, "DCT quantization matricies visually optimized for individual images," *in Proc. of SPIE Conference on Human Vision, Visual Processing and Digital Display IV*, pp. 202-216, 1992.

[123] Wingsoft Corporation. http://www.wingsoft.com

[124] R. B. Wolfgang and E. J. Delp, "A watermark for digital images," *in Proc. Int. Conf. Image Processing*, pp. 219-222, Sept 1996.

[125] X. G. Xia, C. G. Boncelet, and G. R. Arce, "A multiresolution watermark for digital images," *in IEEE ICIP*, vol. 1, pp.548-551, July 1997.

[126] XML. 2001. "Extensible Markup Language (XML) 1.0 (Second Edition)," http://www.w3.org/TR/REC-xml, February 2001.

[127] M. Yeung, and B.L. Yeo, "Fragile Watermarking of 3D Objects," *in International Conference on Image Processing*, 1998.

[128] J. Zhao and E. Koch, "Embedding robust labels into images for copyright protection," *in Proc. of the Int. Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Technologies*, pp.242-251, August 1995.

[129] D. Zheng, Y. Liu, and Jiying Zhao, "RST invariant digital image watermarking based on a new phase-only filtering method," *in Proc. of 7th International Conf. on Signal Processing*, August 2004.

[130] D. Zheng, J. Zhao, and A. El Saddik, "RST-invariant digital image watermarking based on log-polar mapping and phase correlation," *in IEEE Trans. On Circuits and Systems for Video Technology*, vol. 13, pp. 753-765, August 2003.

Appendix

Review of Applied Standards and Techniques

In this appendix, all the necessary background information on MPEG-4 needed for the understanding of the research work done in this area is provided. A brief overview on SMIL is also provided. An overview of Pseudo-random Sequences (PRS) is also explained in this appendix. The PRS were used for the protection of MPEG-4 and SMIL scenes as explained in chapter 4 and chapter 5.

A.1. MPEG-4 Overview

A.1.1. General

MPEG-4 is an ISO/IEC standard for the coding of audio-visual objects. It was developed by the Moving Picture Experts Group (MPEG), the committee that also developed the MPEG-1 and MPEG-2 standards for the coding of moving pictures and their associated audio. MPEG-4 became an International Standard in the first months of 1999, and was called MPEG-4 Version 1. The fully backward compatible extensions under the title of MPEG-4 Version 2 were frozen at the end of 1999, to acquire the formal International Standard Status early in 2000. Several extensions were added since and work on some specific items is still in progress.

A.1.2. MPEG-4 Focus

MPEG-4 was originally intended for very high compression coding of audio-visual information at very low bit-rates of 64 kbits/sec or under. When MPEG-4 video encoding techniques were investigated, it was expected that a scheme capable of achieving very high compression would emerge for standardization. By mid 1994, it was realized that the new video coding schemes were offering only moderate increase in compression, by a factor of 1.5 or so, compared to the already existing techniques. In addition, a new class of multimedia applications was emerging that required increasing levels of functionality than those provided by any other video standard at bit rates ranging from 10kbits/sec to 1024kbits/sec.

These developments lead to broadening the original scope of MPEG-4 to larger range of bit-rates and important new functionalities. MPEG-4 was aimed at providing new audio-visual coding solutions to allow interactivity, universal accessibility and sufficient compression.

A.1.3. MPEG-4 Features

The MPEG-4 standard is rich in advanced features that explore almost every possibility in a digital environment. It provides standardized ways to represent units of aural, visual, or audio-visual content called "media objects". These media objects can be on natural or synthetic origin. They are organized in a hierarchical fashion. At the leaves of this hierarchy are the primitive media objects, such as, images (i.e fixed background), video objects (i.e. a talking person without the background), and audio objects (i.e. voice associated with the person, background music). In addition to these primitive media objects, MPEG-4 defines the coded representation of objects such as: text and graphics, talking synthetic heads and associated text used to synthesize the speech and animate the head, animated bodies to go with the faces, and synthetic sound. All these natural and synthetic objects can be represented in 2D or 3D space.

All the media objects are represented individually independent of their surrounding or background. Some of the primitive media objects can be combined to form compound media objects, which encompass an entire subtree. In Figure A-1 for example, the visual object corresponding to the talking person and the corresponding voice are tied together to form a new compound media object, containing both the aural and visual components of that talking person. Such grouping allows authors to construct complex scenes, and enables consumers to manipulate meaningful sets of objects. The description of those scenes is defined using a binary stream called BInary Format for Scene (BIFS). It is a standardized way for describing the scene in a binary format. More details on BIFS will follow in the next section.

MPEG-4 provides a way to multiplex streams, as well as to synchronize their presentation at the terminal. Multiplexing is achieved using the FlexMux tool, a tool that can multiplex streams, if the underlying transport protocol does not support multiplexing, or because the overhead is very expensive. Synchronization is achieved at the Synchronization layer, where elementary streams are packetized and time stamped to enable their synchronized composition and presentation. MPEG-4 does not define a specific transport protocol. Existed transport formats can be used, like Real-time Transport Protocol (RTP) or MPEG-2 Transport Stream [47][26]. The first is more appropriate for transmission over the internet, while the former is more suitable for digital broadcasting.

A delivery framework called Delivery Multimedia Integration Framework (DMIF) is also provided in MPEG-4. DMIF provides three types of abstraction. The first is the abstraction of the application from the delivery type. Interactive (client-server), local, broadcast or multicast delivery is controlled and accessed through a unique API. The choice of the appropriate delivery mechanism is based on the parsing of the URL that locates the required data or service. Quality of Service (QoS) parameters are also passed to determine the requirements of the application. The second, is an abstraction from the transport protocol, which can be RTP/UDP/IP or AAL5/ATM or MPEG-2 TS or others.

Furthermore, in the interactive or multicast case, DMIF provides an abstraction from the signaling mechanisms of the delivery system.

Moreover, MPEG-4 is designed with features for the protection of intellectual property and digital content, which will be explained in details in section A.1.7.

It should be noted that the above mentioned features are not all the features in MPEG-4, but only a part from which it is of interest for this thesis work. The interested reader can refer to the MPEG-4 standard (ISO/IEC 14496) for more details.

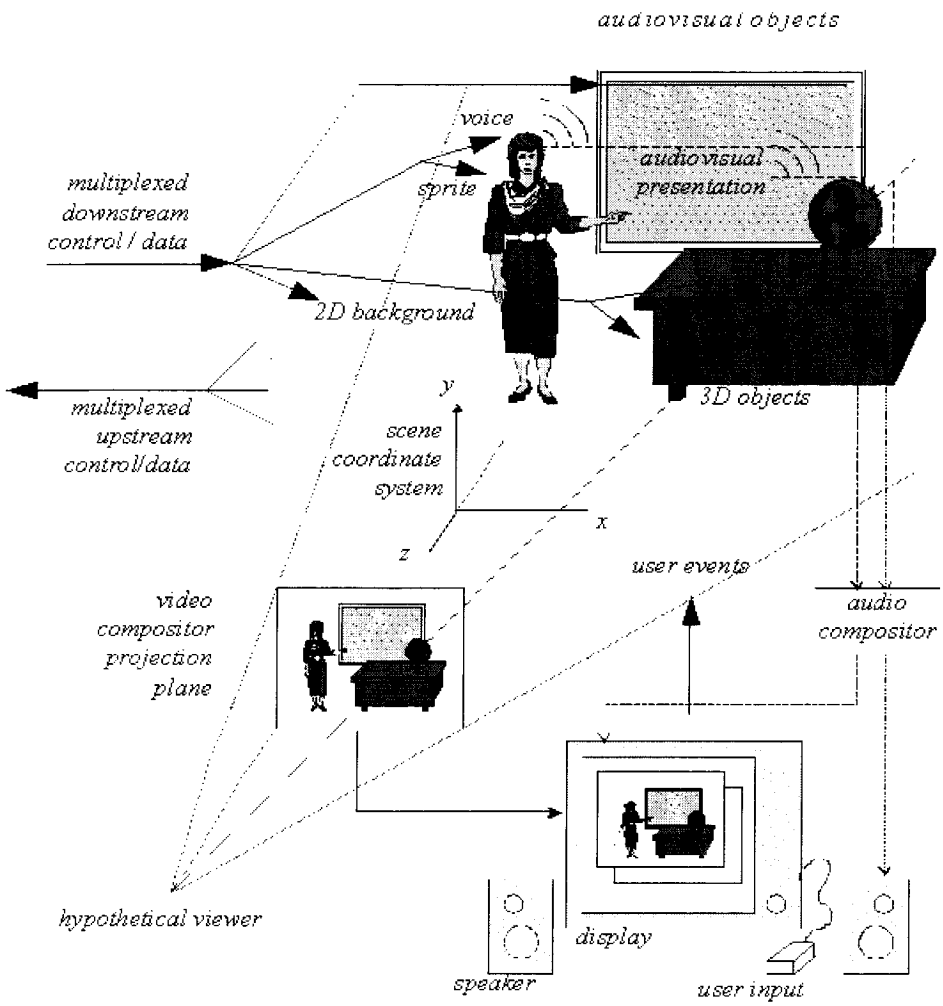


Figure A-1 MPEG-4 scene example

A.1.4. MPEG-4 Standard Organization

The MPEG-4 standard ISO/IEC 14496 consists of the following parts:

ISO/IEC 14496-1: Systems, which standardizes the development of techniques for multiplexing/de-multiplexing, composition and presentation of moving images, audio, graphics and data [48].

ISO/IEC 14496-2: Visual (Natural and Synthetic Video), which standardizes the development of coded representation of moving pictures of natural and synthetic origin, by cooperation with the Synthetic and Natural Hybrid Coding (SNHC) [49].

ISO/IEC 14496-3: Audio (Natural and Synthetic Audio), which standardizes the development of coded representation of audio of natural and synthetic origin, by cooperating with the SNHC [50].

ISO/IEC 14496-4: Conformance, which provides methods for subjective and objective assessment and conducts tests [51].

ISO/IEC 14496-5: Software, evaluates the realization of the coding techniques [52].

ISO/IEC 14496-6: DMIF [53], which standardizes the interfaces between digital storage media, networks, servers and clients for delivery bit-streams in networked environments.

A.1.5. MPEG-4 BInary Format for Scene (BIFS) Representation

MPEG of ISO/IEC has put a standard for the description of 3D scenes called BInary Format for Scenes (BIFS) as part of its MPEG-4 standard [47][48]. BIFS is a compact binary format representing a predefined set of audio-visual objects, their behaviors, and their spatio-temporal relationships.

A.1.5.1. Scene Representation

MPEG-4 uses the same concepts used in VRML for the scene graph consisting of scenes, nodes, fields, and events. MPEG-4 and VRML scenes are both composed of a collection

of nodes, arranged in a direct acyclic graph. Like VRML, these scenes contain the same structuring elements, including Transforms, Grouping nodes, and Bindable nodes. MPEG-4 also contains the same "active" nodes such as Interpolators, Sensors, and of course ROUTEs. However, MPEG-4 also defines many nodes for additional functionality beyond what VRML provides -- MPEG-4 has roughly 100 nodes that fall into 7 categories. Most of the additional nodes support new functionality for 2D, audio, streaming capability, and some system functions (these are described in more detail in a section below).

A.1.5.2. Compression and Binary Representation

An MPEG-4 scene description is sent in a binary format defined by MPEG-4 BIFS. BIFS describes a format for three different aspects of scene description -- scene loading, scene update, and scene animation. These are termed BIFS-Scene, BIFS-Command, and BIFS-Anim respectively. Scene loading and scene update are actually almost identical in that the initial scene loading is structured like one large scene update. As in VRML, fields of nodes are initialized with default values if their values are not specified in a scene description.

The binary format for MPEG-4 scene description attempts to balance several considerations. Compression is, of course, the ultimate goal, but compression conflicts with extensibility, ease of parsing, and a simple specification. The BIFS protocol is a compromise between these goals. Despite that it is a compromise, however, it is still quite an efficient representation.

For compression, BIFS uses a compact representation for the scene components. For example, when a scene parameter is specified, the minimal number of bits needed to distinguish that parameter from others is used. This scheme is used for specifying the scene contents also. For example, there are 18 different components that are used to

specify the geometry of a scene: Spheres, Cones, Polygons, etc. These are indexed and specified using 5 bits.

The actual parameter values associated with the scene parameters have a different quantization scheme, that is actually part of the scene and thus under the control of the scene author. By default, scene parameter values are not quantized at all; they are stored in their native format (i.e. 32 bits for floats and integers, 1 bit for Booleans, etc.). The scene parameters are classified into different categories, and the values in each category can be linearly quantized using quantization parameters (maximum and minimum values, along with the number of quantization bits) that are specified locally in the scene. The categories consist of parameters that should have similar values, i.e. scaling values, 3D coordinate values, etc. This scheme balances the utility of having local quantization control with the cost of specifying the quantization parameters. Note that being able to specify these quantization parameters within the scene provides the author with great control over both the quality and delivery efficiency of the content.

BIFS uses a different quantization scheme in the case of animation. The BIFS-Anim format describes how scene values should be modified over time. The scene author specifies which node parameters should be animated by associating these parameters with a stream of input values, represented by a BIFS-Anim stream. The stream consists of a sequence of initial values (Intra Frames) and successive difference values (P-frames) that are arithmetically encoded. The stream also contains a "mask" which describes how the portions of the stream are decompressed into the various field values to which they are targeted. Note that although this scheme is more complex than the quantization scheme for the scene described above, it allows highly efficient encoding of animation values. This is important because these parameters are often numerous and animate with a very high frame rate, hence the total volume of data can be quite large without efficient compression. Typical values in these cases are: a factor of 3-5 using BIFS with no quantization; 10-15 with quantization; 10-20 using the optimal quantization of version 2; and about 10-30 with combined mesh compression.

A.1.6. Extensible MPEG-4 Textual Format (XMT)

The Extensible MPEG-4 Textual Format (XMT) is a framework for representing MPEG-4 Systems and media content [56]. It became part of the MPEG standard in December 2001. XMT is a textual representation using the eXtensible Markup Language (XML). It allows content authors to exchange their content with other authors, tools, or service providers and facilitates interoperability with MPEG-4, Extensible 3D (X3D) (a standard developed by the Web3D Consortium to replace the Virtual Reality Modeling Language, VRML), and the Synchronized Multimedia Integration Language (SMIL) [107][108]. In this case, there can be an interaction between MPEG-4 players, VRML players, and SMIL players as shown in Figure A-2. MPEG-7 [81] is now being integrated with XMT, which will enable content-based retrieval of MPEG-4 objects. XMT consists of two levels: the XMT-Ω format and the XMT-A format.

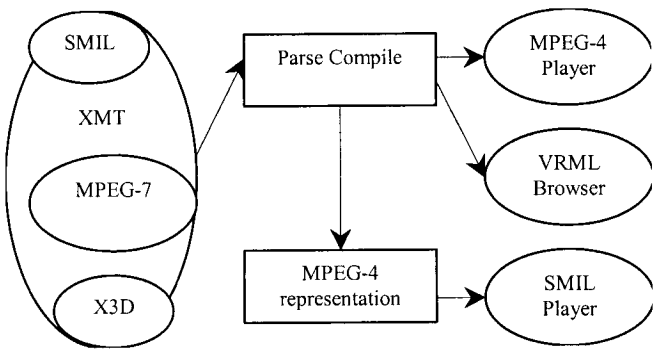


Figure A-2 Interoperability of XMT with other standards [121]

A.1.6.1. XMT-Ω Format

The XMT-Ω is a high level abstraction of MPEG-4 features based on the Synchronized Multimedia Integration Language (SMIL). The SMIL language contains several functional areas, such as Timing and Synchronization, Transitions, and Media that are divided into one or more XML modules. A subset of the modules defined by SMIL are used as a basis of the XMT-Ω format. In addition, a new set of elements that expresses a

high level view of MPEG-4 content has been designed for XMT-Ω. In converting the XMT-Ω format to MPEG-4, there is not necessarily only one mapping for each construct. MPEG-4 nodes and routes are very powerful tools, and there can often be more than one way to represent XMT-Ω constructs. Moreover, recognizing that some authors may want to access low-level nodes and routes, XMT-Ω allows the embedding of the XMT-A node and route definitions within an identified low-level escape section to create custom media constructs.

The XMT-Ω format can be processed and played directly by a SMIL player, or can be converted to the corresponding X3D nodes and played back by a VRML/X3D player, or compiled to MPEG-4 representation and played by an MPEG-4 player. A detailed explanation of the XMT-Ω format can be found in [56].

A.1.6.2. XMT-A Format

The XMT-A format provides a direct textual representation (XML based) of MPEG-4 Systems BIFS and Object Descriptor Framework, including Object Descriptors (ODs) and commands, Object Content Information (OCI) events and commands, and Intellectual Property Management and Protection (IPMP) messages and commands. The ODs associate scene description components with the Elementary Streams that contain the corresponding coded data. These include visual streams and audio streams, as well as animation streams that update elements of the scene more efficiently than do BIFS commands for complex animations. XMT-A also includes a textual representation to allow MPEG-J streams to be created from Java classes or zip files. XMT-A is interoperable with X3D. Such compatibility facilitates content interchange and interoperability.

The following example provides an idea on how XMT-A format representation for a 3D model would look like. Figure A-3 is the view of a 3D box. Its equivalent representation in VRML and XMT is shown in Figure A-4 and Figure A-5 respectively. A XMT-A

document instance has a single <Body> element instance. Inside the <Body> element lies the element <par begin="0.0"> (as shown in Figure A-5), which is one of SMIL time containers to express timing, in this case at time zero. The rest of the XMT-A document is a direct textual mapping of MPEG-4 BIFS.

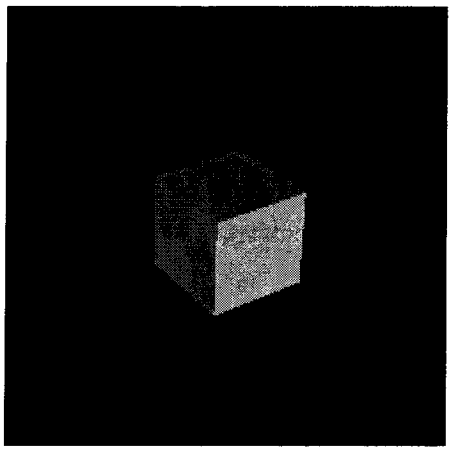


Figure A-3 A simple 3D Box model

```
#VRML V2.0 utf8
Group {
children [
DEF box Transform {
translation 0 1 0
 children [
   Shape {
      appearance Appearance {
            material Material {
                  diffuseColor 0.0 1.0 1.0
            }
      }
      geometry Box {
                  size 0.5 0.5 0.5
      }
   }
 ]} ]}
```

Figure A-4 VRML representation of the 3D box model

```
<Body>
 <par begin="0.0">
  <Replace>
   <Scene>
    <Group  DEF="x5">
     <children>
      <Group>
       <children>
        <Transform  DEF="x10" translation=" 0.0 1.0 0.0">
         <children>
          <Shape>
           <appearance>
            <Appearance>
             <material>
             <Material diffuseColor="0.0 1.0 1.0">
             </Material>
            </material>
           </Appearance>
          </appearance>
          <geometry>
           <Box size="0.5 0.5 0.5">
           </Box>
          </geometry>
         </Shape>
        </children>
       </Transform>
      </children>
     </Group>
    </children>
   </Group>
  </Scene>
 </Replace>
</par>
</Body>
```

Figure A-5 XMT-A representation of the 3D Box model

A.1.7. Intellectual Property Management and Protection (IPMP)

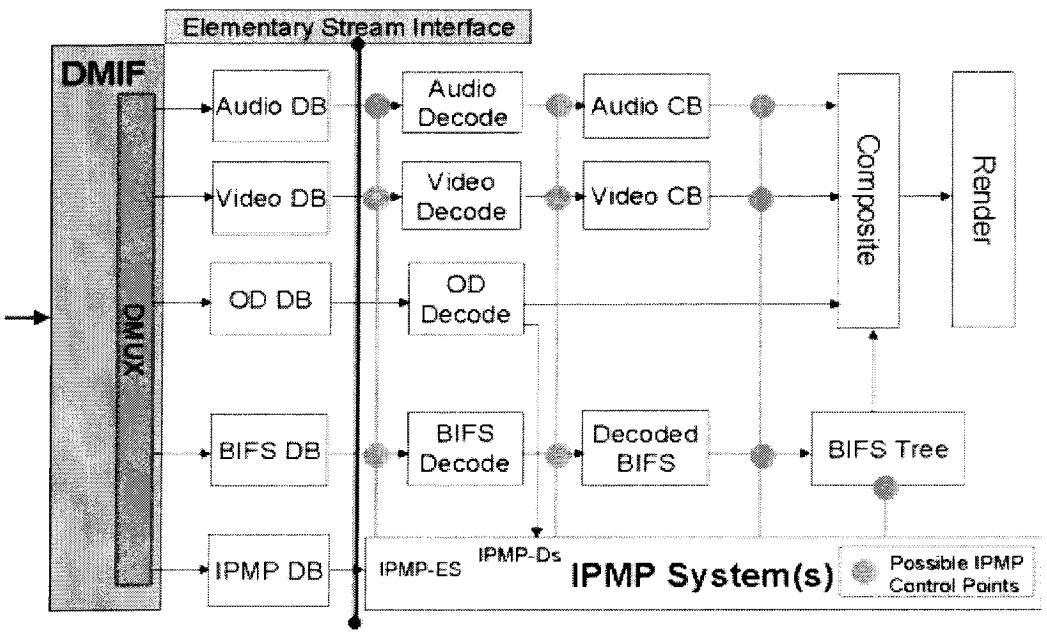
In MPEG-4, the Intellectual Property Management and Protection (IPMP) framework has been standardized [70]. The MPEG-4 standard has been setup for designers and builders of a wide variety of multimedia applications. Each of these applications has a set of requirements regarding protection of the information it manages. These applications can produce conflicting content management and protection requirements. The design of the IPMP standard considers the complexity of the MPEG-4 standard and the diversity of applications. The level and type of protection considered depends on the content's value, complexity, and the sophistication of the associated business models.

While MPEG-4 does not standardize IPMP systems, it does standardize the MPEG-4 IPMP interface. This interface consists of IPMP-Descriptors (IPMP-Ds) and IPMP-Elementary Streams (IPMP-ES).
IPMP-Ds are part of the MPEG-4 object descriptors that describe how an object can be accessed and decoded. These IPMP-Ds are used to denote the IPMP system that was used to encrypt the object. An independent registration authority (RA) is used so any party can register its own IPMP system and identify this without collisions. All MPEG objects are represented by elementary streams, that can reference each other. These special elementary streams (IPMP-ES) can be used to convey IPMP specific data. Their syntax and semantics are not specified in the standard. IPMP-Ds and IPMP-ESs provide a communication mechanism between IPMP systems and the MPEG-4 terminal. Certain applications may require multiple IPMP systems. When MPEG-4 objects require management and protection, they have IPMP-Ds associated with them. These IPMP-Ds indicate which IPMP systems are to be used and provide information to these systems about how to manage and protect the content.

Figure A-6 indicates a variety of "hooks" (control points) in the MPEG-4 terminal at which one might desire IPMP control. Many systems apply control between

demultiplexing and the elementary stream decoders. There are also systems that need to apply control after stream decoding. For example, retrieval of watermarks introduced prior to content encoding can only be done after content decoding. In general the IPMP control points involve different kinds of mechanisms ranging from rule processing to decryption to watermarking. The actual processing of this control occurs in the IPMP system.

Besides enabling owners of intellectual property to manage and protect their assets, MPEG-4 provides a mechanism to identify those assets via the *Intellectual Property Identification Data Set* (IPI Data Set). The IPI Data Set identifies content either by means of internationally standardized numbering systems, such as International Standard Audio-Visual Number (ISAN), ISBN, Digital Object Identifier (DOI), or by privately generated key/value pairs (e.g. »Composer«/»John Lennon«). The IPI Data Set can be used by IPMP systems as input to the management and protection process. For example, this can be used to generate audit trails that track content use.



CB = Composition buffer, DB = Decoding buffer

Figure A-6 Integration of IPMP with MPEG-4 System

A.2. SMIL Overview

The Synchronized Multimedia Integration language (SMIL) is an XML based language developed by the World Wide Web Consortium (W3C). It allows authors to write interactive multimedia presentations. Using SMIL, authors can describe the temporal behavior of media objects (audio, video, image and text) within a multimedia presentation, associate hyperlinks with media objects, describe the layout of the presentation on the screen, and display media following user preferences, language and/or bit-rate.

A number of different players support SMIL scene presentations such as RealOne, GriNS, and Internet Explorer.

Some of the features used in SMIL are:

- Media streaming located on different servers: A multimedia presentation can be put together using media objects located on different servers.
- Layout of a presentation: Using SMIL, various media objects can be arranged in the appropriate positions in the presentation.
- Time and control of a presentation: SMIL provides powerful timing features, that manage the presentation timeline, and keep the media objects rigidly synchronized.
- Stream different presentations to different audiences: SMIL allows one to stream different media clips to different audiences based on criteria such as language preference or available bandwidth. This allows an author to create multiple presentations, while having only one link on his/her Website. When a viewer clicks on the link, the player reads the options in the SMIL file and chooses the appropriate presentation.

- Link to Websites: SMIL's extensive hyperlinking capabilities allow the linking of a streaming presentation to other streaming clips, or Websites. Websites can display automatically at any time during the presentation, or may load only when the viewer clicks on a link.

Please refer to [107] and [108] for more detailed information.

A.3. Pseudo-Random Sequences

Pseudo-Random Sequences (PRS), also called Pseudo-Noise (PN) sequences, or maximal-length shift register sequences (m-sequences), are finite-length sequences generated by a Linear Feedback Shift Register (LFSR).

Pseudo-random sequences coding have been very efficient and used in many areas of communication, cryptology and position recovery. It became widely used after Shannon proved that the one-time-pad used in cryptography is unbreakable. Now they are used in many areas such as error-correcting codes in communication, absolute position detection and coding for mobile robots, and various fields in cryptography.

Because the theory of pseudo-random sequences is based on the theory of finite fields, a short overview of finite fields is given first.

A.3.1. Finite (Galois) Fields

A field is a set of elements **F** equipped with two binary operations "+" (addition) and "*" (multiplication) such that for all a, b, c $\in$ **F** the following axioms hold [16], [34], [65], [75]:

a) **F** is an abelian group under addition ("+"):

   1) Addition in **F** is commutative

$$a+b = b+a;$$

2) Addition in **F** is associative

$$a+(b+c) = (a+b)+c;$$

3) There exists an element *0* belonging to **F** such that

$$a+0 = 0+a = a;$$

for all a belonging to **F**.

4) For each element a belonging to **F**, there exists an element $-a$ belonging to **F** such that

$$a+(-a) = (-a)+a = 0;$$

b) **F** is an abelian group under multiplication ("x"):

1) Multiplication in **F** is commutative

$$a*b = b*a;$$

2) Multiplication in **F** is associative

$$a*(b*c) = (a*b)*c;$$

3) There exists an element *1* belonging to **F** such that:

$$a*1 = 1*a = a;$$

for all a belonging to **F**.

4) For each element a belonging to **F**, there exists an element a^{-1} belonging to **F** such that:

$$a*(a^{-1}) = (a^{-1})*a = 1;$$

c) The following distributive laws hold:

1) $(a+b)*c = (a*c)+(b*c)$;

2) $a*(b+c) = (a*b)+(a*c)$;

The symbol *0* is the identity element of the additive group, and the symbol *1* is the identity element of the multiplicative group.

The rational numbers **Q**, real numbers **R**, and complex numbers **C** are examples of fields. Because each one of these fields contains an infinite number of elements, they are called *infinite fields*. A field that contains a finite number of elements is called a *finite field* [49]. The number of elements in a field is called the order of the field. A field with p elements is called a *Galois field* and is denoted by *GF(p)* [7] [10].

The finite fields are of two types: finite fields of order of a prime number denoted by *GF(p)* and finite fields of order power of a prime number denoted *GF(q)*. For the rest of this writing, unless otherwise noted, p will represent a prime number, and q will represent a power of a prime number.

A.3.2. Finite Fields of Order of Prime Number – GF(p)

The simplest class of finite fields is this with prime number of elements. This class of fields is important because any finite field contains a subfield belonging to this class.

A field belonging to this class can be constructed in the following manner:

> Let p be any prime number. The integers under addition and multiplication modulo p form a field *GF(p)* with p elements $\{0, 1, 2, \ldots, p\text{-}1\}$.

Some illustrations of finite fields *GF(p)* of order of a prime with $p = 2$, 3, and 5 respectively are shown in

+	0	1
0	0	1
1	1	0

*	0	1
0	0	0
1	0	1

Table A-1 Addition and multiplication tables for GF(2)

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

*	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Table A-2 Addition and multiplication tables for GF(3)

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

*	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Table A-3 Addition and multiplication tables for GF(5)

An important property of this class of fields is that the elements of the field, except the element *0* (the identity element for addition), form a cyclic group under multiplication. This means that any finite field with a prime number of elements contains at least one element called a generator from which all the other elements of the field can be obtained. For example, the element 2 for GF(5) is a generator element for the cyclic group {1, 2, 3, 4} under multiplication modulo 5:

$$2^1 = 2;\ 2^2 = 4;\ 2^3 = 3;\ 2^4 = 1;$$

A.3.3. Finite Fields of Order Power of Prime Number – GF(p^m)

The class of finite fields that has a number of elements equal to a power of a prime is the most general class of finite fields. These classes of fields are known as polynomial fields, or Galois fields of order $q=p^m$, and are denoted by GF(p^m).

Unlike finite fields GF(p) of order of prime number that are defined by taking the integers modulo p with the operations of addition and multiplication also done modulo p, finite fields GF(p^m) of order power of a prime number are not defined in this straight forward way because integers modulo $q = p^m$ do not form a field. In this case, the nonzero elements are expressed as the first q-1 powers of some primitive element t of the field GF(q). In order to do arithmetic in such a finite field, whose order is power of a prime, it is necessary to have a primitive element and its minimal polynomial. It should be noted that this minimal polynomial is not unique. Addition is done in the normal way, modulo p. While multiplication is done by multiplying modulo p, and taking the remainder when divided by a minimal polynomial $h(x)$.

The minimal polynomial $h(x)$ of a primitive element of GF(p^m) has degree m and it is called a primitive polynomial [72]. This polynomial has coefficients from GF(p) and is irreducible over GF(p). By definition, a polynomial $h(x)$ with coefficients from GF(p) is irreducible over GF(p) if it is not the product of two polynomials of lower degree with coefficients from GF(p).

The irreducible polynomial $h(x)$ used to generate a field is called the generator polynomial. It is proven that for any positive integer m and any prime p, there exists a primitive polynomial of degree m with coefficients from GF(p) [16].

The field constructed in this way contains p^m elements and is denoted by GF(p^m). It is proven that a finite field of order p^m exists for all primes p and positive integers m, and that these are the only finite fields that exist.

Consider the following example:

The field GF(3^3) has $q = p^m = 3^3 = 27$ elements that are polynomials whose coefficients represent all possible three-element combinations of the elements of its prime order subfield GF(3) = {0, 1, 2}.

The elements of the field GF(3^3) in polynomial representation are: 0, 1, 2, x, x+1, x+2, 2x, 2x+1, 2x+2, x^2, x^2+1, x^2+2, x^2+x, x^2+x+1, x^2+x+2, x^2+2x, x^2+2x+1, x^2+2x+2, $2x^2$, $2x^2$+1, $2x^2$+2, $2x^2$+x, $2x^2$+x+1, $2x^2$+x+2, $2x^2$+2x, $2x^2$+2x+1, $2x^2$+2x+2.

The mechanism used to obtain this polynomial representation is summarized in Table A-4.

3-Element combination	Polynomial equivalent	Polynomial representation
000	$0*x^2 + 0*x^1 + 0*x^0$	0
001	$0*x^2 + 0*x^1 + 1*x^0$	1
010	$0*x^2 + 1*x^1 + 0*x^0$	x
100	$1*x^2 + 0*x^1 + 0*x^0$	x^2
012	$0*x^2 + 1*x^1 + 2*x^0$	x+2
120	$1*x^2 + 2*x^1 + 0*x^0$	x^2+2x
212	$2*x^2 + 1*x^1 + 2*x^0$	$2x^2$+x+2
111	$1*x^2 + 1*x^1 + 1*x^0$	x^2+x+1
122	$1*x^2 + 2*x^1 + 2*x^0$	x^2+2x+2
202	$2*x^2 + 0*x^1 + 2*x^0$	$2x^2$+2
011	$0*x^2 + 1*x^1 + 1*x^0$	x+1
110	$1*x^2 + 1*x^1 + 0*x^0$	x^2+x
112	$1*x^2 + 1*x^1 + 2*x^0$	x^2+x+2
102	$1*x^2 + 0*x^1 + 2*x^0$	x^2+2
002	$0*x^2 + 0*x^1 + 2*x^0$	2
020	$0*x^2 + 2*x^1 + 0*x^0$	2x
200	$2*x^2 + 0*x^1 + 0*x^0$	$2x^2$
021	$0*x^2 + 2*x^1 + 1*x^0$	2x+1
210	$2*x^2 + 1*x^1 + 0*x^0$	$2x^2$+x
121	$1*x^2 + 2*x^1 + 1*x^0$	x^2+2x+1
222	$2*x^2 + 2*x^1 + 2*x^0$	$2x^2$+2x+2
211	$2*x^2 + 1*x^1 + 1*x^0$	$2x^2$+x+1
101	$1*x^2 + 0*x^1 + 1*x^0$	x^2+1
022	$0*x^2 + 2*x^1 + 2*x^0$	2x+2
220	$2*x^2 + 2*x^1 + 0*x^0$	$2x^2$+2x
221	$2*x^2 + 2*x^1 + 1*x^0$	$2x^2$+2x+1
201	$2*x^2 + 0*x^1 + 1*x^0$	$2x^2$+1

Table A-4 Elements of the finite field GF(3^3)

The polynomial addition of this field is performed modulo 3:

$(x^2+x+1) + (x^2+2x) = (2x^2+3x+1)$ modulo 3

$= 2x^2+1$

$(x+1) + (2x^2+2x+2) = (2x^2+3x+3)$ modulo 3

$= 2x^2$

On the other hand, polynomial multiplication is performed modulo 3 and taking the remainder when dividing by a primitive polynomial. This statement implies that the polynomial multiplication in a finite field of order power of a prime is not unique and depends always on the primitive polynomial used.

The polynomial multiplication in $GF(3^3)$ when the primitive polynomial is $h(x) = x^3+2x+1$ is as follows:

$(x^2+2x)*(2x^2+1) = (2x^4+4x^3+x^2+2x)$ modulo 3

$= (2x^4+x^3+x^2+2x)$

$= (x^3+2x+1)*(2x+1) + (x+2)$

$= h(x)*(2x+1) + (x+2)$

$= x+2$

The q elements of a finite field $GF(p^m)$ can be expressed in four possible ways:

- as a *m*-tuple of elements belonging to GF(*p*)
- as a polynomial of degree lower or equal to *m* with coefficients from GF(*p*)
- as a power of a primitive element *t* (is zero for h(x))
- as a logarithm of base *t*

The first two representations, as a m-tuple and as a polynomial of degree lower or equal to *m* with coefficients from GF(*p*), are best suited for addition; while the last two representations, as a power of a primitive element *t* and as a logarithm of base *t* are best suited for multiplication [72] [73] [100].

Every finite field contains at least one primitive element from which all the other elements of the field, except the element *0*, can be obtained by multiplication of that primitive element with itself. This implies that the elements $\{1, t, t^2, \ldots, t^{q-2}\}$ belonging to a finite field GF(q} – {0}, where GF(q) = GF(p^m), form a cyclic group under the operation of multiplication and that $t^{q-1} = t^{q-2}*t=1$, $t^q = t^{q-1}*t=1*t=t$, $t^{q+1}=t^2$, etc.

Following are some examples that present the idea on how to find the elements of GF(p^m) of order power of a prime number and construct the addition and multiplication tables for those field types. The finite fields GF(2^2), GF(2^3), and GF(3^2) will be presented.

Finite Field GF(2^2)

Construct GF(2^2) having a primitive polynomial $h(x) = x^2+x+1$

The field GF(2^2) has $q = p^m = 2^2 = 4$ elements that are polynomials whose coefficients represent all possible two-element combinations of the elements of its prime order subfield GF(2) = {0, 1}.

The elements of the finite field GF(2^2) in all representation forms are shown in TableA-6.

The primitive element t is a root for $h(x) = x^2+x+1$. This implies that $h(t) = t^2+t+1 = 0$ or $t^2 = t+1$. Therefore, the equivalency between elements in polynomial form and power of t form is shown in Table A-5 for $h(x) = x^2+x+1$.

This should be noted that the elements {1, t, t^2} of GF(2^2) – {0} form a cyclic group under multiplication. Therefore, $t^3=t^2*t=1$, $t^4=t^3*t=t$, etc.

Power of t representation	Polynomial representation
0	0
1	1
t	x
$t^2=t+1$	x+1

Table A-5 Polynomial & Power of t equivalency of GF(2^2) over $h(x)= x^2+x+1$

2-element comb.	Polynomial	Power of t	Logarithm	Symbol	Natural number
00	0	0	$-\infty$	0	0
01	1	1	0	1	1
10	X	t	1	A	2
11	x+1	t^2	2	B	3

Table A-6 The elements of GF(2^2) generated by h(x) = x^2+x+1

The addition and multiplication of elements of the finite field GF(2^2) is shown in Table A-7 and Table A-8 respectively. All calculations are done modulo 2. Since the primitive element t is a root for h(x) = x^2+x+1, it should be reminded that t^2+t+1 = 0 or t^2 = t+1.

Symbol representation	Polynomial representation
0+δ = δ+0 = δ	δ : represent any element from GF(2^2)
1+1 = 0	(1)+(1) = (2) (modulo 2) = 0
1+A = A+1 = B	(1)+(x) = (x)+(1) = x+1
1+B = B+1 = A	(1)+(x+1) = (x+1)+(1) = (x+2) (modulo 2) = x
A+A = 0	(x)+(x) = (2x) (modulo 2) = 0
A+B = B+A = 1	(x)+(x+1) = (x+1)+(x) = (2x+1) (modulo 2) = 1
B+B = 0	(x+1)+(x+1) = (2x+2) (modulo 2) = 0

Table A-7 Addition of elements of GF(2^2) using polynomial representation

Symbol representation	Power of t representation
0*δ = δ*0 = 0	δ : represent any element from GF(2^2)
1*δ = δ*1 = δ	δ : represent an element from GF(2^2)
A*A = B	(t)*(t) = (t^2)
A*B = B*A = 1	(t)*(t^2) = (t^2)*(t) = t^3 = 1
B*B = A	(t^2)*(t^2) = t^4 = t^3*t = 1*t = t

Table A-8 Multiplication of elements of GF(2^2) using power of t representation

For better readability, instead of using symbol representation {0, 1, A, B}, Table A-9 shows the addition and multiplication of elements over GF(2^2) using natural numbers {0, 1, 2, 3}.

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Table A-9 Addition and multiplication of elements of GF(4)

Finite Field GF(2^3)

Construct GF(2^3) having a primitive polynomial $h(x) = x^3+x+1$

The field GF(2^3) has $q = p^m = 2^3 = 8$ elements that are polynomials whose coefficients represent all possible two-element combinations of the elements of its prime order subfield GF(2) = {0, 1}.

The elements of the finite field GF(2^3) in all representation forms are shown in Table A-11.

The primitive element t is a root for $h(x) = x^3+x+1$. This implies that $h(t) = t^3+t+1 = 0$ or $t^3 = t+1$. Therefore, the equivalency between elements in polynomial form and power of t form is shown in Table A-10 for $h(x) = x^3+x+1$.

This should be noted that the elements {1, t, t^2, t^3, t^4, t^5, t^6} of GF(2^3) – {0} form a cyclic group under multiplication. Therefore, $t^7=t^6*t=1$, $t^8=t^7*t=t$, etc.

Power of t representation	Polynomial representation
0	0
1	1
t	x
t^2	x^2
$t^3=t+1$	x+1
$t^4=t^3*t=(t+1)*t=t^2+t$	x^2+x
$t^5=t^3*t^2=(t+1)*t^2=t^3+t^2=t^2+t+1$	x^2+x+1
$t^6=t^3*t^3=(t+1)*(t+1)=t^2+t+t+1=t^2+1(\text{mod}2)$	x^2+1

Table A-10 Polynomial & Power of t equivalency of GF(2^3) over h(x)= x^3+x+1

3-element comb.	Polynomial	Power of t	Logarithm	Symbol	Natural number
000	0	0	$-\infty$	0	0
001	1	1	0	1	1
010	X	t	1	A	2
100	x^2	t^2	2	B	3
011	x+1	t^3	3	C	4
110	x^2+x	t^4	4	D	5
111	x^2+x+1	t^5	5	E	6
101	x^2+1	t^6	6	F	7

Table A-11 The elements of GF(2^3) generated by h(x) = x^3+x+1

The addition and multiplication of elements of the finite field GF(2^3) is shown in Table A-12 and Table A-13 respectively. All calculations are done modulo 2. Since the primitive element t is a root for h(x) = x^3+x+1, it should be reminded that $t^3+t+1 = 0$ or $t^3 = t+1$.

Symbol representation	Polynomial representation
$0+\delta = \delta+0 = \delta$	δ : represent any element from $GF(2^3)$
$1+1 = 0$	$(1)+(1) = (2)(\text{mod}2) = 0$
$1+A = A+1 = C$	$(1)+(x) = (x)+(1) = x+1$
$1+B = B+1 = F$	$(1)+(x^2) = (x^2)+(1) = x^2+1$
$1+C = C+1 = A$	$(1)+(x+1) = (x+1)+(1) = (x+2)(\text{mod}2) = x$
$1+D = D+1 = E$	$(1)+(x^2+x) = (x^2+x)+(1) = x^2+x+1$
$1+E = E+1 = D$	$(1)+(x^2+x+1) = (x^2+x+1)+(1) = (x^2+x+2)(\text{mod}2) = x^2+x$
$1+F = F+1 = B$	$(1)+(x^2+1) = (x^2+1)+(1) = (x^2+2)(\text{mod}2) = x^2$
$A+A = 0$	$(x)+(x) = (2x)(\text{mod}2) = 0$
$A+B = B+A = D$	$(x)+(x^2) = (x^2)+(x) = x^2+x$
$A+C = C+A = 1$	$(x)+(x+1) = (x+1)+(x) = (2x+1)(\text{mod}2) = 1$
$A+D = D+A = B$	$(x)+(x^2+x) = (x^2+x)+(x) = (x^2+2x)(\text{mod}2) = x^2$
$A+E = E+A = F$	$(x)+(x^2+x+1) = (x^2+x+1)+(x) = (x^2+2x+1)(\text{mod}2) = x^2+1$
$A+F = F+A = E$	$(x)+(x^2+1) = (x^2+1)+(x) = x^2+x+1$
$B+B = 0$	$(x^2)+(x^2) = (2x^2)(\text{mod}2) = 0$
$B+C = C+B = E$	$(x^2)+(x+1) = (x+1)+(x^2) = x^2+x+1$
$B+D = D+B = A$	$(x^2)+(x^2+x) = (x^2+x)+(x^2) = (2x^2+x)(\text{mod}2) = x$
$B+E = E+B = C$	$(x^2)+(x^2+x+1) = (x^2+x+1)+(x^2) = (2x^2+x+1)(\text{mod}2) = x+1$
$B+F = F+B = E$	$(x^2)+(x^2+1) = (x^2+1)+(x^2) = (2x^2+1)(\text{mod}2) = 1$
$C+C = 0$	$(x+1)+(x+1) = (2x+2)(\text{mod}2) = 0$
$C+D = D+C = F$	$(x+1)+(x^2+x) = (x^2+x)+(x+1) = (x^2+2x+1)(\text{mod}2) = x^2+1$
$C+E = E+C = B$	$(x+1)+(x^2+x+1) = (x^2+x+1)+(x+1) = (x^2+2x+2)(\text{mod}2) = x^2$
$C+F = F+C = E$	$(x+1)+(x^2+1) = (x^2+1)+(x+1) = (x^2+x+2)(\text{mod}2) = x^2+x$
$D+D = 0$	$(x^2+x)+(x^2+x) = (2x^2+2x)(\text{mod}2) = 0$
$D+E = E+D = 1$	$(x^2+x)+(x^2+x+1) = (x^2+x+1)+(x^2+x) = (2x^2+2x+1)(\text{mod}2) = 1$
$D+F = F+D = C$	$(x^2+x)+(x^2+1) = (x^2+1)+(x^2+x) = (2x^2+x+1)(\text{mod}2) = x+1$
$E+E = 0$	$(x^2+x+1)+(x^2+x+1) = (2x^2+2x+2)(\text{mod}2) = 0$
$E+F = F+E = A$	$(x^2+x+1)+(x^2+1) = (x^2+1)+(x^2+x+1) = (2x^2+x+2)(\text{mod}2) = x$
$F+F = 0$	$(x^2+1)+(x^2+1) = (2x^2+2)(\text{mod}2) = 0$

Table A-12 Addition of elements of $GF(2^3)$ using polynomial representation

Symbol representation	Power of t representation
$0*\delta = \delta*0 = 0$	δ : represent any element from GF(2^3)
$1*\delta = \delta*1 = \delta$	δ : represent any element from GF(2^3)
$A*A = B$	$(t)*(t) = t^2$
$A*B = B*A = C$	$(t)*(t^2) = (t^2)*(t) = t^3$
$A*C = C*A = 1$	$(t)*(t^3) = (t^3)*(t) = t^4$
$A*D = D*A = 1$	$(t)*(t^4) = (t^4)*(t) = t^5$
$A*E = E*A = 1$	$(t)*(t^5) = (t^5)*(t) = t^6$
$A*F = F*A = 1$	$(t)*(t^6) = (t^6)*(t) = t^7 = 1$
$B*B = D$	$(t^2)*(t^2) = t^4$
$B*C = C*B = E$	$(t^2)*(t^3) = (t^3)*(t^2) = t^5$
$B*D = D*B = F$	$(t^2)*(t^4) = (t^4)*(t^2) = t^6$
$B*E = E*B = 1$	$(t^2)*(t^5) = (t^5)*(t^2) = t^7 = 1$
$B*F = F*B = A$	$(t^2)*(t^6) = (t^6)*(t^2) = t^8 = t$
$C*C = F$	$(t^3)*(t^3) = t^6$
$C*D = D*C = 1$	$(t^3)*(t^4) = (t^4)*(t^3) = t^7 = 1$
$C*E = E*C = A$	$(t^3)*(t^5) = (t^5)*(t^3) = t^8 = t$
$C*F = F*C = B$	$(t^3)*(t^6) = (t^6)*(t^3) = t^9 = t^8*t = t*t = t^2$
$D*D = A$	$(t^4)*(t^4) = t^8 = t^7*t = 1*t = t$
$D*E = E*D = B$	$(t^4)*(t^5) = (t^5)*(t^4) = t^9 = t^8*t = t*t = t^2$
$D*F = F*D = C$	$(t^4)*(t^6) = (t^6)*(t^4) = t^{10} = t^9*t = t^2*t = t^3$
$E*E = C$	$(t^5)*(t^5) = t^{10} = t^9*t = t^2*t = t^3$
$E*F = F*E = D$	$(t^5)*(t^6) = (t^6)*(t^5) = t^{11} = t^{10}*t = t^3*t = t^4$
$F*F = E$	$(t^6)*(t^6) = t^{12} = t^{11}*t = t^4*t = t^5$

Table A-13 Multiplication of elements of GF(2^3) using power of t representation

For better readability, instead of using symbol representation {0, 1, A, B, C, D, E, F}, Table A-14 shows the addition and multiplication of elements over GF(2^3) using natural numbers {0, 1, 2, 3, 4, 5, 6, 7}.

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	4	7	2	6	5	3
2	2	4	0	5	1	3	7	6
3	3	7	5	0	6	2	4	1
4	4	2	1	6	0	7	3	5
5	5	6	3	2	7	0	1	4
6	6	5	7	4	3	1	0	2
7	7	3	6	1	5	4	2	0

*	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	3	4	5	6	7	1
3	0	3	4	5	6	7	1	2
4	0	4	5	6	7	1	2	3
5	0	5	6	7	1	2	3	4
6	0	6	7	1	2	3	4	5
7	0	7	1	2	3	4	5	6

Table A-14 Addition and multiplication of elements of GF(8)

Finite Field GF(3^2)

Construct GF(3^2) having a primitive polynomial $h(x) = x^2+x+2$

The field GF(3^2) has $q = p^m = 3^2 = 9$ elements that are polynomials whose coefficients represent all possible two-element combinations of the elements of its prime order subfield GF(3) = {0, 1, 2}.

The elements of the finite field GF(3^2) in all representation forms are shown in Table A-16.

The primitive element t is a root for $h(x) = x^2+x+2$. This implies that $h(t) = t^2+t+2 = 0$ or $t^2 = (-t-2)(\text{modulo } 3) = 2t+1$ ➔ $t^2 = 2t+1$. Therefore, the equivalency between elements in polynomial form and power of t form is shown in Table A-15 for $h(x) = x^2+x+2$.

This should be noted that the elements {1, t, t^2, t^3, t^4, t^5, t^6, t^7} of GF(3^2) – {0} form a cyclic group under multiplication. Therefore, $t^8=t^7*t=1$, $t^9=t^8*t=t$, etc.

Power of t representation	Polynomial representation
0	0
1	1
T	x
$t^2=2t+1$	2x+1
$t^3=t^2*t=(2t+1)*t=2t^2+t=2(2t+1)+t=(5t+2)(mod3)=2t+2$	2x+2
$t^4=t^3*t=(2t+2)*t=2t^2+2t=2(2t+1)+2t=(6t+2)(mod3)=2$	2
$t^5=t^4*t=2t$	2x
$t^6=t^5*t=(2t)*(t)=2t^2=2(2t+1)=(4t+2)(mod3)=t+2$	x+2
$t^7=t^6*t=(t+2)*(t)=t^2+2t=(2t+1)+2t=(4t+1)(mod3)=t+1$	x+1

Table A-15 Polynomial & Power of t equivalency of GF(3^2) over h(x)= x^2+x+2

2-element comb.	Polynomial	Power of t	Logarithm	Symbol	Natural number
00	0	0	$-\infty$	0	0
01	1	1	0	1	1
10	X	t	1	A	2
21	2x+1	t^2	2	B	3
22	2x+2	t^3	3	C	4
02	2	t^4	4	D	5
20	2x	t^5	5	E	6
12	x+2	t^6	6	F	7
11	x+1	t^7	7	G	8

Table A-16 The elements of GF(3^2) generated by h(x) = x^2+x+2

The addition and multiplication of elements of the finite field GF(3^2) is shown in Table A-17 and Table A-18 respectively. All calculations are done modulo 3. Since the primitive element t is a root for h(x) = x^2+x+2, it should be reminded that $t^2+t+2 = 0$ or $t^2 = 2t+1$.

Symbol representation	Polynomial representation
$0+\delta = \delta+0 = \delta$	δ : represent any element from $GF(3^2)$
1+1 = D	(1)+(1) = 2
1+A = A+1 = G	(1)+(x) = (x)+(1) = x+1
1+B = B+1 = C	(1)+(2x+1) = (2x+1)+(1) = 2x+2
1+C = C+1 = E	(1)+(2x+2) = (2x+2)+(1) = 2x
1+D = D+1 = 0	(1)+(2) = (2)+(1) = (3)(mod3) = 0
1+E = E+1 = B	(1)+(2x) = (2x)+(1) = 2x+1
1+F = F+1 = A	(1)+(x+2) = (x+2)+(1) = (x+3)(mod3) = x
1+G = G+1 = F	(1)+(x+1) = (x+1)+(1) = x+2
A+A = E	(x)+(x) = 2x
A+B = B+A = 1	(x)+(2x+1) = (2x+1)+(x) = (3x+1)(mod3) = 1
A+C = C+A = D	(x)+(2x+2) = (2x+2)+(x) = (3x+2)(mod3) = 2
A+D = D+A = F	(x)+(2) = (2)+(x) = x+2
A+E = E+A = 0	(x)+(2x) = (2x)+(x) = (3x)(mod3) = 0
A+F = F+A = C	(x)+(x+2) = (x+2)+(x) = 2x+2
A+G = G+A = B	(x)+(x+1) = (x+1)+(x) = 2x+1
B+B = F	(2x+1)+(2x+1) = (4x+2)(mod3) = x+2
B+C = C+B = A	(2x+1)+(2x+2) = (2x+2)+(2x+1) = (4x+3)(mod3) = x
B+D = D+B = E	(2x+1)+(2) = (2)+(2x+1) = (2x+3)(mod3) = 2x
B+E = E+B = G	(2x+1)+(2x) = (2x)+(2x+1) = (4x+1)(mod3) = x+1
B+F = F+B = 0	(2x+1)+(x+2) = (x+2)+(2x+1) = (3x+3)(mod3) = 0
B+G = G+B = D	(2x+1)+(x+1) = (x+1)+(2x+1) = (3x+2)(mod3) = 2
C+C = G	(2x+2)+(2x+2) = (4x+4)(mod3) = x+1
C+D = D+C = B	(2x+2)+(2) = (2)+(2x+2) = (2x+4)(mod3) = 2x+1
C+E = E+C = F	(2x+2)+(2x) = (2x)+(2x+2) = (4x+2)(mod3) = x+2
C+F = F+C = 1	(2x+2)+(x+2) = (x+2)+(2x+2) = (3x+4)(mod3) = 1
C+G = G+C = 0	(2x+2)+(x+1) = (x+1)+(2x+2) = (3x+3)(mod3) = 0
D+D = 1	(2)+(2) = (4)(mod3) = 1
D+E = E+D = C	(2)+(2x) = (2x)+(2) = 2x+2
D+F = F+D = G	(2)+(x+2) = (x+2)+(2) = (x+4)(mod3) = x+1
D+G = G+D = A	(2)+(x+1) = (x+1)+(2) = (x+3)(mod3) = x
E+E = A	(2x)+(2x) = (4x)(mod3) = x
E+F = F+E = D	(2x)+(x+2) = (x+2)+(2x) = (3x+2)(mod3) = 2
E+G = G+E = 1	(2x)+(x+1) = (x+1)+(2x) = (3x+1)(mod3) = 1
F+F = B	(x+2)+(x+2) = (2x+4)(mod3) = 2x+1
F+G = G+F = E	(x+2)+(x+1) = (x+1)+(x+2) = (2x+3)(mod3) = 2x
G+G = C	(x+1)+(x+1) = 2x+2

Table A-17 Addition of elements of $GF(3^2)$ using polynomial representation

Symbol representation	Power of t representation
$0*\delta = \delta*0 = 0$	δ : represent any element from $GF(3^2)$
$1*\delta = \delta*1 = \delta$	δ : represent any element from $GF(3^2)$
A*A = B	$(t)*(t) = t^2$
A*B = B*A = C	$(t)*(t^2) = (t^2)*(t) = t^3$
A*C = C*A = D	$(t)*(t^3) = (t^3)*(t) = t^4$
A*D = D*A = E	$(t)*(t^4) = (t^4)*(t) = t^5$
A*E = E*A = F	$(t)*(t^5) = (t^5)*(t) = t^6$
A*F = F*A = G	$(t)*(t^6) = (t^6)*(t) = t^7$
A*G = G*A = 1	$(t)*(t^7) = (t^7)*(t) = t^8 = 1$
B*B = D	$(t^2)*(t^2) = t^4$
B*C = C*B = E	$(t^2)*(t^3) = (t^3)*(t^2) = t^5$
B*D = D*B = F	$(t^2)*(t^4) = (t^4)*(t^2) = t^6$
B*E = E*B = G	$(t^2)*(t^5) = (t^5)*(t^2) = t^7$
B*F = F*B = 1	$(t^2)*(t^6) = (t^6)*(t^2) = t^8 = 1$
B*G = G*B = A	$(t^2)*(t^7) = (t^7)*(t^2) = t^9 = t^8*t = 1*t = t$
C*C = F	$(t^3)*(t^3) = t^6$
C*D = D*C = G	$(t^3)*(t^4) = (t^4)*(t^3) = t^7$
C*E = E*C = 1	$(t^3)*(t^5) = (t^5)*(t^3) = t^8 = 1$
C*F = F*C = A	$(t^3)*(t^6) = (t^6)*(t^3) = t^9 = t^8*t = 1*t = t$
C*G = G*C = B	$(t^3)*(t^7) = (t^7)*(t^3) = t^{10} = t^9*t = t*t = t^2$
D*D = 1	$(t^4)*(t^4) = t^8 = 1$
D*E = E*D = A	$(t^4)*(t^5) = (t^5)*(t^4) = t^9 = t^8*t = 1*t = t$
D*F = F*D = B	$(t^4)*(t^6) = (t^6)*(t^4) = t^{10} = t^9*t = t*t = t^2$
D*G = G*D = C	$(t^4)*(t^7) = (t^7)*(t^4) = t^{11} = t^{10}*t = t^2*t = t^3$
E*E = B	$(t^5)*(t^5) = t^{10} = t^8*t^2 = 1*t^2 = t^2$
E*F = F*E = C	$(t^5)*(t^6) = (t^6)*(t^5) = t^{11} = t^{10}*t = t^2*t = t^3$
E*G = G*E = D	$(t^5)*(t^7) = (t^7)*(t^5) = t^{12} = t^{11}*t = t^3*t = t^4$
F*F = D	$(t^6)*(t^6) = t^{12} = t^8*t^4 = 1*t^4 = t^4$
F*G = G*F = E	$(t^6)*(t^7) = (t^7)*(t^6) = t^{13} = t^{12}*t = t^4*t = t^5$
G*G = F	$(t^7)*(t^7) = t^{14} = t^8*t^6 = 1*t^6 = t^6$

Table A-18 Multiplication of elements of $GF(3^2)$ using power of t representation

For better readability, instead of using symbol representation {0, 1, A, B, C, D, E, F, G}, Table A-19 shows the addition and multiplication of elements over $GF(3^2)$ using natural numbers {0, 1, 2, 3, 4, 5, 6, 7, 8}.

+	0	1	2	3	4	5	6	7	8
0	0	1	2	3	4	5	6	7	8
1	1	5	3	8	7	0	4	6	2
2	2	3	6	4	1	8	0	5	7
3	3	8	4	7	5	2	1	0	6
4	4	7	1	5	8	6	3	2	0
5	5	0	8	2	6	1	7	4	3
6	6	4	0	1	3	7	2	8	5
7	7	6	5	0	2	4	8	3	1
8	8	2	7	6	0	3	5	1	4

*	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8
2	0	2	3	4	5	6	7	8	1
3	0	3	4	5	6	7	8	1	2
4	0	4	5	6	7	8	1	2	3
5	0	5	6	7	8	1	2	3	4
6	0	6	7	8	1	2	3	4	5
7	0	7	8	1	2	3	4	5	6
8	0	8	1	2	3	4	5	6	7

Table A-19 Addition and multiplication of elements of GF(9)

A.3.4. Pseudo-Random Sequence Generation

Pseudo-random sequences (PRS), which are also called pseudo-noise (PN), maximal-length shift-register sequences, or m-sequences [4] [18][23][31][40][99], are sequentially generated as the output of a shift register.

For any given primitive polynomial $h(x)$ of degree m with coefficients from GF(q), a feedback shift register can be constructed as follows:

- the number of memory cells is equal to m
- each memory cell contains an element of the field GF(q)
- at each time unit, the contents of the memory cells are shifted one place to the right
- the content of the left most memory element is the sum of the terms corresponding to $h(x)$ (all operations are done in GF(q)).

Considering an irreducible polynomial

$$h(x) = x^m + h_{m-1}x^{m-1} + \ldots + h_1x + h_0$$

with $h_i \in GF(q)$, $h_0 \neq 0$. The Linear Feedback Shift Register (LFSR), shown in Figure A-7, is specified by h(x)=0 and $x^m = -h_{m-1}x^{m-1} - h_{m-2}x^{m-2} - \ldots - h_1x - h_0$.

In this figure the cells contain elements of GF(q), say a_{i+m-1}, ..., a_i. The feedback path then forms

$$a_{i+m} = -h_{m-1}a_{i+m-1} - h_{m-2}a_{i+m-2} - \ldots - h_1a_{i+1} - h_0a_i$$

which is the recurrence describing the output sequence. This is an infinite sequence of period p^m-1 (if the starting state is not zero), and each nonzero state appears once in a period. A segment of the output sequence of length p^m-1 is called a pseudo-random sequence over GF(q).

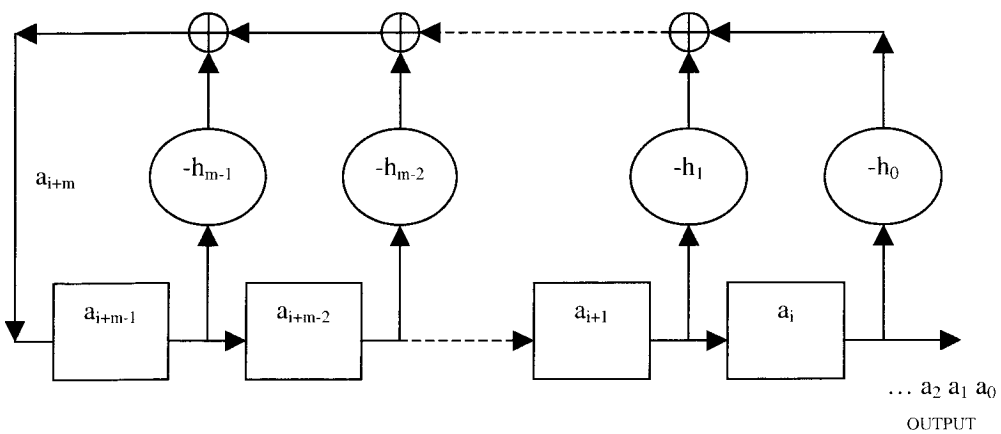


Figure A-7 Linear Feedback Shift Register specified by h(x)

For each finite field GF(q) there are q^m-1 PRSs that are basically the same sequence shifted to the right or left by a number of symbols depending on the state of the register when the sequence generation was started.

Since there are m memory elements containing numbers between 0 and $q-1$, there are q^m possible states for the shift register. Thus, the infinite sequence that the shift register will generate is periodic. The state in which all the memory elements are zero is not included because the shift register cannot leave this state. Hence, the maximum length of the sequence is q^m-1. This is the reason why the pseudo-random sequences are also called maximum-length shift register sequences.

In general, given a finite field GF(q), where q is a prime or power of a prime number, a pseudo-random sequence is generated by a m-position LFSR having a feedback path specified by a primitive polynomial of degree m with coefficients from the Galois field GF(q). This sequence is called Pseudo-Random Multi-Valued Sequence (PRMVS) because it is constructed from multi-valued entries taken from the elements of GF(q), with $q > 2$. Whereas, a Pseudo-Random Binary Sequence (PRBS), is a binary sequence generated from the two elements of GF(q) with $q = 2$.

In order to generate Pseudo-Random Sequences (PRSs) over a finite field GF(q), it is necessary to have the arithmetic operations (addition and multiplication tables described in section A.3.2) defined for that field along with an irreducible polynomial h(x) of degree m used to implement the LFSR that will generate the PRS. As mentioned previously, the maximum length that can be generated by LFSR is q^m-1.

A.3.5. Properties of Pseudo-Random Sequences

Let $h(x)$ be a fixed primitive polynomial of degree m, and let $\delta_m(q)$ be the set consisting of the pseudo-random sequences obtained from $h(x)$. The following properties of these sequences apply [72]:

Property 1. The Shift Property: If $b=b_0b_1\ldots b_{q-2}$ is any PRS in $\delta_m(q)$, then any cyclic shift in b, say

$$b_jb_{j+1}\ldots b_{q-2}b_0\ldots b_{j-1}$$

is also in $\delta_m(q)$.

Property 2. The Recurrence Property: Any PRS $b\in \delta_m(q)$ satisfies the recurrence

$$b_{i+m} = h_{m-1}b_{i+m-1}+h_{m-2}b_{i+m-2}+\ldots+h_1b_{i+1}+b_i \qquad i = 0, 1, \ldots$$

Property 3. The Window Property: If a window of width m is slid along a PRS in $\delta_m(q)$, each of the p^m-1 non-zero binary m-tuple is seen exactly once. This follows from the fact that h(x) is a primitive polynomial. Let us consider the PRS generated by the primitive polynomial $h(x) = x^3+2x+1$ over GF(3). A cycle of this sequence is given by:

00101211**201**110020212210222

The three bolded symbols form a window that is unique in this sequence.

Note: to avoid any difficulties at the ends of the sequence, imagine that the PRS is written in a circle. That is, after the last digit in the sequence comes the first digit in the sequence.

Property 4. The Pseudo-Random Property: In any PRS in $\delta_m(q)$, *0* occurs q^{m-1}-1 times and every non-zero element of GF(q) occurs q^{m-1} times.

Property 5. The Addition Property: The sum of two sequences in $\delta_m(q)$ (formed component-wise, modulo m, without carries) is another sequence in $\delta_m(q)$.

Property 6. The Shift-and-Add Property: The sum of a pseudo-random sequence and a cyclic shift of itself is another pseudo-random sequence.

Property 7a: A PRS in $\delta_m(q)$ has the form

$$a=b,\ \gamma b,\ \gamma^2 b,\ \ldots,\ \gamma^{q-2} b,$$

where b is a sequence of length $(q^m-1)/(q-1)$ and γ is a primitive element of GF(q). This is because the states of the shift register can be made to correspond to a logarithmic table of GF(q).

For example, if $q = 2^2$, $m = 2$ and $h(x) = x^2+x+w$, the following pseudo-random sequence is obtained:

$$011w^2 10ww1w0w^2w^2ww^2$$

In this case $b= 011w^2 1$ and $\gamma = w$.

Property 7b: Let $a = (a_0 \ldots a_{n-1})$ be a PRS in $\delta_m(q)$, and let $b =(a_s a_{s+1} \ldots a_{s-1}) = (b_0 \ldots b_{n-1})$ be a shift of a by s places. i) If s is not a multiple of q-1, then among the q^m-1 pairs (a_i,b_i), (0, 0) occurs q^{m-2}-1 times and every other pair of elements of GF(q) occurs q^{m-2} times; ii) If $s = j(q-1)$, then (0, 0) occurs q^{m-1}-1 times and the pairs $(\alpha, \gamma^j\alpha)$, for all nonzero α in GF(q), occur q^{m-1} times. In particular, if s = 0 (no shift) each (α, α), $\alpha \neq 0$, occurs q^{m-1} times. For example, if the PRS given in property 7a is shifted once,

thus

$$011w^2 10ww1w0w^2w^2ww^2$$
$$11w^2 10ww1w0w^2w^2ww^20$$

then (0, 0) does not occur and every other pair (0, 1), (0, w), ..., (w^2, w^2) occurs once.

A.3.6. Examples of Pseudo-Random Sequences

The following are some examples that illustrate how to generate PRSs over finite fields. For the sake of space, the examples will be given for the finite fields GF(2), GF(3), GF(4), and GF(5) only. For other finite fields, they can be generated in a similar manner.

PRS of 4^{th} Degree over GF(2)

The generation of PRS generated by an irreducible polynomial $h(x) = x^4 + x^3 + 1$ of degree $m = 4$ defined over the finite field GF(2), whose operation tables are presented in Table A-1, is shown in Figure A-8.

The feedback equation for the LFSR that will generate the PRS results from the condition

$$h(x) = x^4 + x^3 + 1 = 0$$

which is equivalent to

$$x^4 = -x^3 - 1$$
$$= x^3 + 1$$

because $x^3 + x^3 = 0$ (modulo 2) $\Leftrightarrow x^3 = -x^3$ for all $x \in GF(2)$

The 4-cell LFSR will generate a PRS of maximum length $n = 2^4-1 = 15$. It has a feedback equation represented by $a_{k+4} = a_{k+3} + a_k$.

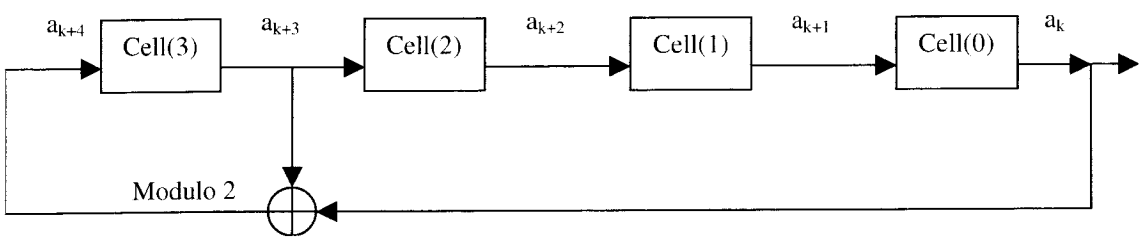


Figure A-8 LFSR over $h(x) = x^4+x^3+1$ in GF(2)

Having a seed value s = 0001, the following PRBS will be generated and is cyclic after length = 15.

$$PRS = 000111101011001$$

$$Length = 15$$

PRS of 3rd Degree over GF(3)

The generation of PRS generated by an irreducible polynomial $h(x) = x^3 + 2x + 1$ of degree m = 3 defined over the finite field GF(3), whose operation tables are presented in Table A-2, is shown in Figure A-9.

The feedback equation for the LFSR that will generate the PRS results from the condition

$$h(x) = x^3 + 2x + 1 = 0$$

which is equivalent to

$$x^3 = -2x - 1$$

$$= x + 2$$

because $2x + x = 0$ (modulo 3) $\Leftrightarrow x = -2x$ and $1+2 = 0$ (modulo 3) $\Leftrightarrow 2 = -1$ in GF(3)

The 3-cell LFSR will generate a PRS of maximum length $n = 3^3-1 = 26$. It has a feedback equation represented by $a_{k+3} = a_{k+1} + 2a_k$.

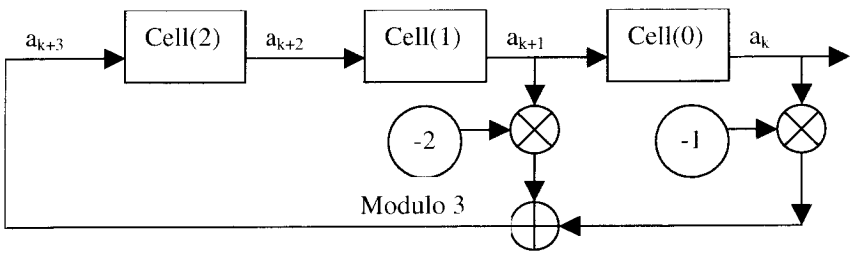


Figure A-9 LFSR over $h(x) = x^3+2x+1$ in GF(3)

Having a seed value s = 001, the following PRMVS will be generated and is cyclic after a period of length = 26.

PRS = 00101211201110020212210222

Length = 26

PRS of 3rd Degree over GF(4)

The generation of PRS generated by an irreducible polynomial $h(x) = x^3 + 3x^2 + x + 2$ of degree m = 3 defined over the finite field $GF(p^m) = GF(2^2)$, whose operation tables generated by the primitive polynomial $g(x) = x^2+x+1$ are presented in Table A-9, is shown in Figure A-10.

The feedback equation for the LFSR that will generate the PRS results from the condition

$$h(x) = x^3 + 3x^2 + x + 2 = 0$$

which is equivalent to

$$x^3 = -3x^2 - x - 2$$
$$= x^2 + 3x + 2$$

because $x^2 + 3x^2 = 0$ (modulo 4) $\Leftrightarrow$ $x^2 = -3x^2$, $x + 3x = 0$ (modulo 4) $\Leftrightarrow$ $3x = -x$, and $2+2 = 0$ (modulo 4) $\Leftrightarrow$ $2 = -2$ in GF(4)

The 3-cell LFSR will generate a PRS of maximum length $n = 4^3-1 = 63$. It has a feedback equation represented by $a_{k+3} = a_{k+2} + 3a_{k+1} + 2a_k$.

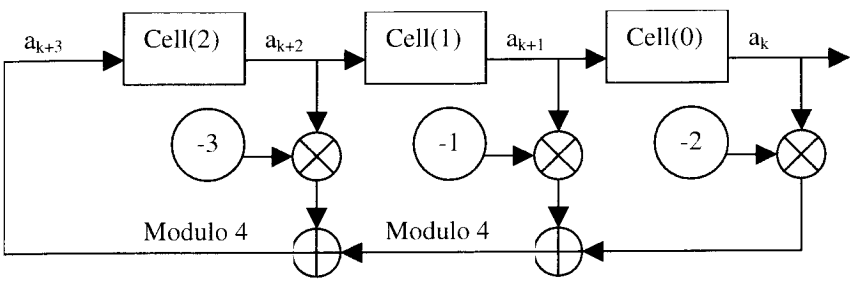


Figure A-10 LFSR over $h(x) = x^3+3x^2+x+2$ in GF(4)

Having a seed value s = 001, the following PRMVS will be generated and is cyclic after a period of length = 63.

PRS = 00133302022332313203100211103033113121301200322201011221232102 3

Length = 63

PRS of 2nd Degree over GF(5)

The generation of PRS generated by an irreducible polynomial $h(x) = x^2 + x + 2$ of degree m = 2 defined over the finite field GF(5), whose operation tables are presented in Table A-3, is shown in Figure A-11.

The feedback equation for the LFSR that will generate the PRS results from the condition

$$h(x) = x^2 + x + 2 = 0$$

which is equivalent to

$$x^2 = -x - 2$$

$$= 4x + 3$$

because $4x + x = 0$ (modulo 5) $\Leftrightarrow 4x = -x$ and $3+2 = 0$ (modulo 5) $\Leftrightarrow 3 = -2$ in GF(5)

The 2-cell LFSR will generate a PRS of maximum length $n = 5^2-1 = 24$. It has a feedback equation represented by $a_{k+2} = 4a_{k+1} + 3a_k$.

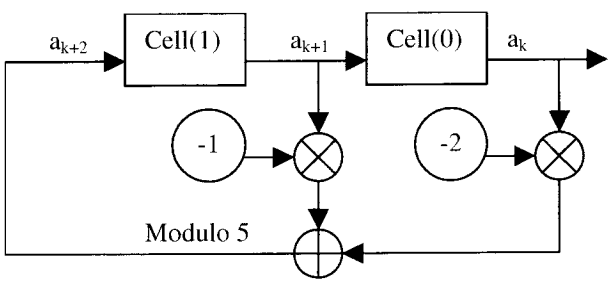


Figure A-11 LFSR over $h(x) = x^2+x+2$ in GF(5)

Having a seed value s = 01, the following PRMVS will be generated and is cyclic after a period of length = 24.

PRS = 014434023313041121032242

Length = 24

A.3.7. Pseudo-Random Window Index Recovery

Having an irreducible polynomial *h(x)* of degree *m* with coefficients from GF(*p*), a PRS can be generated using a feedback shift register as explained earlier. Consider a m-size window belonging to this sequence, represented using bold characters below, where the element a_k is considered to be the origin of the window.

$$a = a_0 a_1 a_2 \ldots \mathbf{a_k a_{k+1} a_{k+2} \ldots a_{k+m-1}} a_{k+m} \ldots a_{q-2}$$

The problem of window index recovery can be stated as follows:

> *Given the elements of an m-size window belonging to a PRS generated by h(x) over GF($q=p^m$) determine the k index of the origin of the window* a_k.

Due to the fact that a window represents an element of GF(*q*) in its *m*-tuple form, the problem can be stated as determining the logarithm representation of a given element of the field when the *m*-tuple form is known.

This translation is always necessary for practical applications that use PRSs for encoding. Few methods known in literature that can be applied for solving that problem:

- A serial-type code conversion algorithm, extensively discussed in [117] exploits the reversibility of the PRS generating algorithm. This method is based on the idea that it is possible to find the index (the logarithm representation) associated

with any pseudo-random m-tuple by simply counting the number of reverse feedback shifts that it takes for the given m-tuple to shift back into the initial state of the shift register.

- A strictly parallel solution would be to use a code conversion table stored in ROM. This is expensive for applications requiring high encoding resolutions.
- A compromise [88] is a combination of the serial and parallel. This method employs a parallel-type code conversion where only selected PRS windows, called milestones, are saved in ROM; and also a serial-type code conversion where feedback shifts are counted forward to get to one of the saved milestones.