Forensic audio watermarking for digital video broadcasting

Sun, Y.

Award date:
2009

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computer Science

Forensic Audio Watermarking for Digital Video Broadcasting

by

Yuanyuan Sun

Supervisor:

Dr.D.S.Jarnikov

August 2009

Abstract

With the advancement of the communication networks and devices, it is much easier to deliver high-quality digital multimedia content to people all around the world. At the same time, the advancement makes it challenging for content owners to protect their products from illegal distribution. Currently, how to effectively protect the multimedia content from illegal distribution has become an important technological issue.

Digital watermarking technology has been proposed as a promising solution for the illegal distribution problem. It allows content owners to embed forensic watermark into the original digital content. Once an infringement occurs, the watermark extracted from a pirated copy can be used to track the pirate.

This thesis proposes two forensic watermarking application techniques for digital video broadcasting (DVB): server-guided watermarking and set-top box watermarking. Both techniques allow applying audio watermarking algorithm in the compressed domain and can be used in forensic tracking.

Acknowledgements

The research related to this thesis has been carried out at Irdeto R&D department, Eindhoven, Netherlands. Hereby, I would like to express my appreciation to all of my colleagues for their great support on my project. In particular, I would like to thank my supervisor, Dr. Dmitri Jarnikov, for his professional guidance and valuable advice.

Contents

List of Figures

List of Tables

Abbreviations

AAC	Advanced Audio Coding
A/D	Analog-to-digital converter
AC-3	Dolby AC-3 audio coding system
ATSC	Advanced Television Systems Committee
bps	Bits Per Second
CA	Conditional Access
CAS	Conditional Access System
CAT	Conditional Access Table
CD	Compact Disc
CE	Consumer Electronics
CRC	Cyclic Redundancy Check
DMB-T/H	Digital Multimedia Broadcasting-Terrestrial/Handheld
DTS	Decoding Time Stamp
DVB	Digital Video Broadcasting
DVD	Digital Versatile Disc
ES	Elementary Stream
ESCR	Elementary Stream Clock Reference
HAS	Human Auditory System
ISDB	Integrated Services Digital Broadcasting
ISO	International Organization for Standardization
JTAG	Joint Test Action Group
MP2	MPEG-1 Layer II
MPEG	Moving Pictures Experts Group
NIT	Network Information Table
PAT	Program Association Table

PC	Personal Computer
PCI	Peripheral Component Interconnect
PCR	Program Clock Reference
PDA	Personal Digital Assistant
PES	Packetized Elementary Stream
PID	Packet Identifier
PMT	Program Map Table
PSI	Program Specific Information
PTS	Presentation Time Stamp
STB	Set-top Box
SCFSI	Scale Factor Selection Information
TS	Transport Stream
VOD	Video-on-demand

Chapter 1

Introduction

Recent years have seen the rapid development of digital multimedia communications and the rapid growth in the availability of multimedia content in a digital format (video, audio and images). Consumer electronic (CE) devices (e.g. personal computers, PDAs, CD and DVD recorders, digital video cameras, etc) have made it more convenient for users to get and exchange a large amount of high-quality digital multimedia data. The development at the same time poses more challenges for content owners to protect their multimedia products from illegal distribution. Digital television (TV) as a main media of multimedia distribution suffers from the illegal distribution problem. A large amount of illegal copies of TV programs have been put on internet for download and viewing. Thus, a real demand to find ways for protection against illegal distribution has been increased.

1.1 Content protection approaches

The conventional approach for content protection is to use cryptographic techniques. In cryptography, encryption/scrambling is the process of transforming the original information (*plaintext*) into encrypted/scrambled information (*ciphertext*) by using an algorithm (*cipher*). The detailed operation of a cipher is controlled both by the algorithm and a secret piece of information, called a *key*. The encrypted/scrambled information is unreadable to anyone except those possessing a proper *key*. Without a proper key, an eavesdropper would be unable to decrypt/descramble the message, even

if the encryption method was known. Decryption/descrambling is the reverse process of encryption/scrambling, in other words, moving from the unintelligible ciphertext back to plaintext [1]. CE devices often use conditional access (CA) or digital rights management (DRM) system to protect the multimedia content from unauthorized access or transfer. Both of the two content protection systems contain cryptographic components. The principle idea of CA/DRM is that the protected digital multimedia content is encrypted before the transmission and only the client devices with the appropriate entitlements have the key to decrypt the content. Unfortunately, for these solutions once the information has been fully decrypted, it can be illegally redistributed without any obstacles.

Another approach for content protection is digital watermarking. This technology allows content owners and broadcasters to embed some information into original digital content such as video, audio or image. The embedded information is called digital watermark. Figure 1.1 gives an overview of general watermarking system.

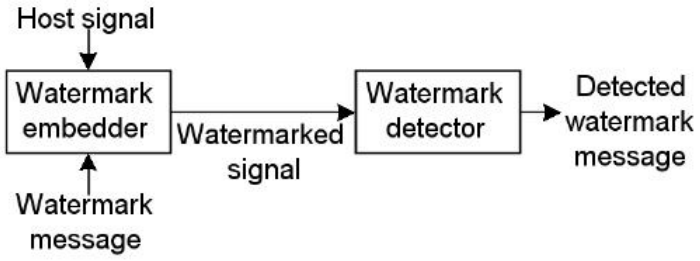

FIGURE 1.1: A general watermarking system [2, Fig.1.1]

In this system, the watermark embedder has two inputs and one output. The two inputs are the watermark message and the host signal (e.g. image, video clip and audio sequence, etc). The output is watermarked signal. Inside this watermark embedder, the watermark message is inserted into original multimedia content and then the watermarked signal is sent out. Later, this watermarked signal can be presented to the watermark detector and the watermark message can be extracted by the detector.

A possible countermeasure against illegal distribution problem is to do forensic tracking based on watermarking technology. Forensic tracking aims at identifying the pirate (the unauthorized distributor) when an illegal copy is found and possibly to prosecute him. In order to track the pirate, watermark embedder embeds unique and traceable information

(such as a user ID), called forensic watermark, into video or/and audio streams, so that content owner can find the pirates by analyzing the watermark message extracted from the pirated copy. Watermarking technology used for forensic tracking is called forensic watermarking. It cannot block all the actions that may lead to an infringement, but it can track the unjust user and be used as a deterrent to piracy.

1.2 Digital TV broadcasting

Digital TV broadcasting is a technology of broadcasting and receiving digital TV signals. The government-issued guidelines that dictate this technology is called digital TV broadcasting standard. There are four sets of worldwide digital TV broadcasting standards [3]:

- DVB (Digital Video Broadcasting), adopted by most countries in Europe, Asia, Africa and Oceania.
- ATSC (Advanced Television Systems Committee), adopted by the USA, Canada, Mexico and South Korea;
- ISDB (Integrated Services Digital Broadcasting), adopted by Japan;
- DMB-T/H (Digital Multimedia Broadcasting-Terrestrial/Handheld), adopted by China (including Hong Kong and Macao);

Because our watermarking solutions are designed for the European digital TV system, DVB is chosen as the broadcasting standard in this project.

All the features and constraints of DVB's coding system should adhere to are described in DVB-MPEG [4]. In the encoding system, the audio data is operated in a compressed domain. The audio compression technology is used to reduce the size of the audio content. There are two main categories of audio compression: lossy compression and lossless compression. Lossless compression eliminates redundancy from the audio content such that no information is lost. Lossy compression eliminates part of original information that almost cannot be perceived by the human auditory system to achieve a specific compression ratio (that is, the size of the compressed file compared to that of the uncompressed file) or bit rate. However, there is a trade-off between the compression

ratio/bit rate and audio quality; relatively high compression ratios or low bit rate can be achieved at the expense of audio quality. DVB-MPEG specifies that the audio encoding should conform to one of the following four lossy audio compression standards: MPEG-1 audio [5], MPEG-2 multichannel audio [6], AAC (Advanced Audio Coding) [7] or AC-3 [8]. The first three standards are MPEG audio compression standards. MPEG (Moving Picture Experts Group) is an international organization which sets standards for audio and video compression and transmission. MPEG-1 and MPEG-2 are two MPEG standards. Every MPEG standard consists of different parts. Each part covers a certain specification. For example, MPEG-1 audio is the third part of MPEG-1 standard. MPEG-2 multichannel audio and AAC are the third and the seventh part of MPEG-2 standard respectively. Furthermore, both MPEG-1 audio and MPEG-2 multichannel audio are divided into 3 layers. Depending on the application, different layers of the coding system with increasing encoder complexity and performance can be used [5]. AC-3 (Dolby Digital) is the marketing name for a series of audio compression technologies developed by Dolby Laboratories. Among these audio compression formats, the use of MPEG-1 Layer II (MP2) is recommended in DVB-MPEG. Besides, DVB-MPEG [4] also specifies that the encoding of DVB system is based on MPEG-2 System [9] which is the first part of MPEG-2 standard. In DVB systems, data can be distributed using a variety of approaches, such as by satellite (DVB-S), cable (DVB-C), terrestrial transmission (DVB-T), etc [3].

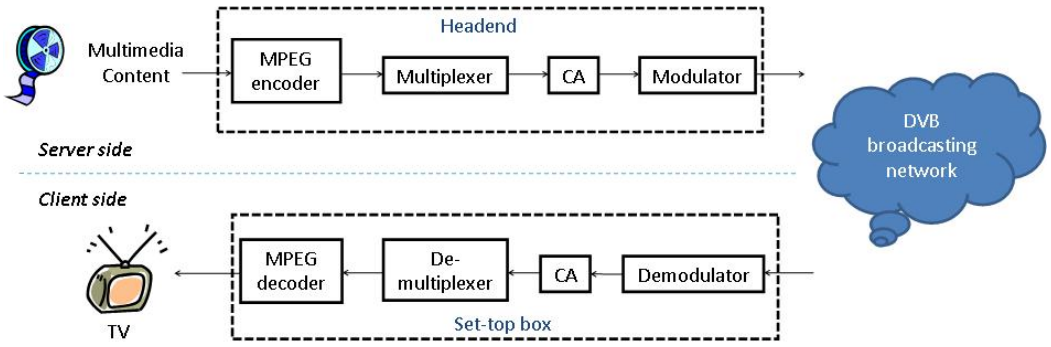

FIGURE 1.2: Digital TV distribution system

Figure 1.2 shows a modern digital TV distribution system which uses DVB as the broadcasting standard. The headend at the server side converts the media sources into a form that can be broadcasted through the DVB broadcasting network. Client devices receive the signal and turn it into programme content which is then displayed on the screen.

Conditional Access (CA) module at the headend scrambles the media source and the one at the client side performs real-time descrambling.

1.3 Watermark embedding approaches

According to embedding place in broadcasting system, the watermarking technology can be conventionally divided into two groups:

- *Server-side watermarking*

  A watermark identifying content owner or provider is embedded at the headend. This method doesn't use any processing resources of the client device. However, since the embedded watermark message cannot uniquely identify the client device, it cannot be applied in forensic tracking.

- *Set-top boc watermarking*

  A forensic watermark is embedded at the set-top box (STB). This method can be used for forensic tracking and doesn't affect the server. However, it imposes all the computational cost of watermark embedding on the STB, so probably the STB doesn't have sufficient processing resources to perform the real-time watermarking. Furthermore, there is a potential danger that the watermarking algorithm may be exposed which renders the watermark vulnerable to counter measures. Civolution's VTrack and Thomson's NexGuard are two examples of commercial deployment of this method.

In order to overcome the problems of the two conventional methods mentioned above, two hybrid methods have currently been designed and developed. Both of the two hybrid methods can be used for forensic tracking and, compared to the set-top box watermarking, they take up less processing resources of the client device.

- *Server-guided watermarking*

  This method embeds watermarks at the headend and makes the watermarked stream unique at the client side. The application at the client side bears lower computational complexity than that of the set-top watermarking and it does not expose the watermarking algorithm. However, this method may reduce the bit

rate of the audio stream depending on the number of data to be watermarked and hence may cause the drop of the audio quality.

- *Set-top box watermarking with pre-processing*

  This method divides the watermarking process into two steps: analysis and insertion. The analysis process is performed at the server as a pre-processing. The pre-processing involves signal processing of audio/video data to determine the locations at which the unique watermark should be inserted in the content [10]. The actual watermark insertion process is performed at the client device. Compared to the set-top box watermarking, the method eases the watermarking process at client side, but it consumes extra bandwidth to send the analyzing result. Cinea's Runing Mark and Widevine's Mensor are two examples of commercial deployment of this approach.

In fact, the two conventional methods can also be used in other distribution systems such as video-on-demand (VoD) and download service. The basic principle of VoD is that users have access to a large catalogue of film and TV titles from which to select content to view at a time of their choice [11]. For VoD and download service, the server-side watermarking can directly use the subscriber serial number or transaction code as the forensic watermark to be embedded into files that are requested by users. However, this method requires lots of processing resources of the server. If a large amount of files are requested at the same time, it is possible that the method leads to a certain delay before the viewing or download starts. While the set-top box watermarking moves the computational cost for watermark embedding to the client device. In this case, the watermarking algorithm may be exposed to the hacker, which is a risk to the security of this algorithm. One possible solution to this is to leave the watermarking algorithm on the server and let the users machine do preparatory computations. This leads to strategies for client-server watermarking [12].

Furthermore, the watermarking schemes can be classified into two groups depending on whether the watermark is embedded to compressed data or raw data.

1. *Uncompressed domain watermarking*

   This method can operate with all compression formats, because it presupposes that the multimedia content is uncompressed to embed watermark and then recompress

again [13]. However, if the content is stored and transmitted in a compressed format, the audio data have to go through the three-step (decoding-watermarking-encoding) process which is very resource-consuming and takes a risk of signal quality degradation.

2. *Compressed domain watermarking*

   This method is preferred when the data is stored and transmitted in a compressed format. However, it only targets a specific compression format. If the data to be watermarked is compressed in other formats, the watermarking algorithm should be transformed to operate with corresponding format.

Because the multimedia content is always carried in a compressed format to the server, the compressed domain watermarking is a good choice for server-side watermarking and server-guided watermarking if the watermarking scheme can operate with the same compressed format. For set-top box watermarking, either watermarking scheme can be chosen. The compressed domain watermarking can operate the data before the decoding at the client side, or the uncompressed domain watermarking can be applied after the decoding. The previous method has low complexity, but the watermark has to go through the decoding procedure which may cause the robustness problem; while the latter method is more robust but it needs more computational resources.

1.4 Project goals

The global goal of this project is to design and implement two watermark embedding solutions for DVB. One is a server-guided watermarking solution and the other is a set-top box watermarking solution as introduced in section 1.3. In both solutions, the watermarks can be detected for forensic tracking and the watermarking algorithm operates on the audio data in the compressed domain. The two solutions should meet the following requirements:

- The embedding algorithm cannot cause perceptible audio artifacts.
- The embedding algorithm should be performed on compressed audio data.
- The computational cost of the embedding algorithm should be within the processing ability of the devices at which the embedding algorithm is performed.

The embedding algorithm includes all the data manipulation in order to apply watermarking in the digital TV broadcasting system. The principle design challenge for these two embedding solutions is the manipulation of the compressed audio streams and integration of the existing watermarking algorithm with other applications to meet the above memtioned requirements. Moreover, for STB watermarking, the limited processing resources of set-top box is also a big challenge.

1.5 Overview

Chapter 2 describes basic principles of DVB, which are helpful to understand the remaining chapters. The global design of two solutions is described in Chapter 3. This chapter also includes a simple description of the watermarking algorithm used in this project. The feasibility analysis, implementation and test for the two watermarking solutions are discussed in Chapter 4. In the last chapter, conlusions and future work are described.

Chapter 2

MPEG and DVB basics

This chapter describes the basic encoding technology of DVB audio and system.

2.1 DVB audio

As mentioned in section 1.2, the use of MPEG-1 Layer II (MP2) as the audio compression format is recommended in DVB-MPEG. Hence, in this project, the audio data is compressed complying with MP2. The MP2 audio stream consists of frames. A frame is the smallest unit to be decoded. Each audio frame contains a 4-byte header at its beginning followed by possibly 16-bit CRC (error check code), audio data and ancillary data. The audio data section contains four sub-sections: bit allocation, scale factor selection information (SCFSI), scale factor and sub-band samples. All of the four sub-sections have a variable length. The encoding technology for MP2 audio data is described in ISO/IEC 11172-3 [5]. The audio frame format is illustrated in Figure 2.1.

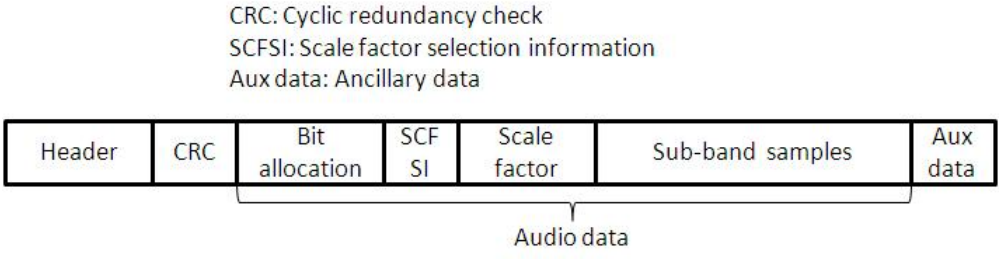

FIGURE 2.1: MPEG-1 Layer II audio frame format

The 4-byte frame header holds the basic structural information about the bit stream such as synchronisation word (sync word), ID, layer, protection bit, bit rate, sampling frequency, padding bit, private bit, mode extension, and etc. The detail about the header format and its meaning are described in Appendix A. The frame itself consists of an integer number of bytes and the number of bytes for each audio frame is indicated in the frame header:

$$N = 144 * bit\_rate/sampling\_frequency. \tag{2.1}$$

If the calculation does not give an integer number, the result is truncated and padding bit specified in the frame header is required. Since the bit rate and sampling frequency are both constant for MP2 audio stream, the length of audio frame is constant as well.

For MP2, the audio signal is split into 32 sub-bands and each one has three groups of samples with 12 samples per group, so the total number of samples in one audio frame is 1152. The bit allocation section specifies the word lengths (in bits) for encoding the samples belonging to a sub-band. The length of each field in bit allocation section depends on the mode extension, data rate and sampling frequency in the frame header. The number of bits used for each allocation field is indicated in LAYER II BIT ALLOCATION TABLES [5]. The SCFSI section contains 2-bit scale factor selection information fields. The number of SCFSI fields depends on the value of allocation fields. If the value of allocation field for one sub-band is not zero, the corresponding SCFSI field for the same sub-band exists. Otherwise, the SCFSI field doesn't exist. The 2-bit SCFSI field indicates whether one, two or three scale factors are transmitted for each sub-band, and how they are applied. For the detail of this process, refer to [5]. The scale factor section consists of 6-bit scale factors. The sample part contains data from 1152 sub-band samples. The ancillary bits are user definable and of a varying length.

2.2 DVB System

As mentioned in section 1.2, MPEG-2 system is used as the coding technology of DVB system. In MPEG-2 system, compressed data from one or more programs is stored and tranmitted in a single stream called transport stream (TS).

2.2.1 Multiplexing

Multiplexing is a process where multiple data streams are combined into one signal over a shared medium. The main principle of multiplexing is illustrated in Figure 2.2.

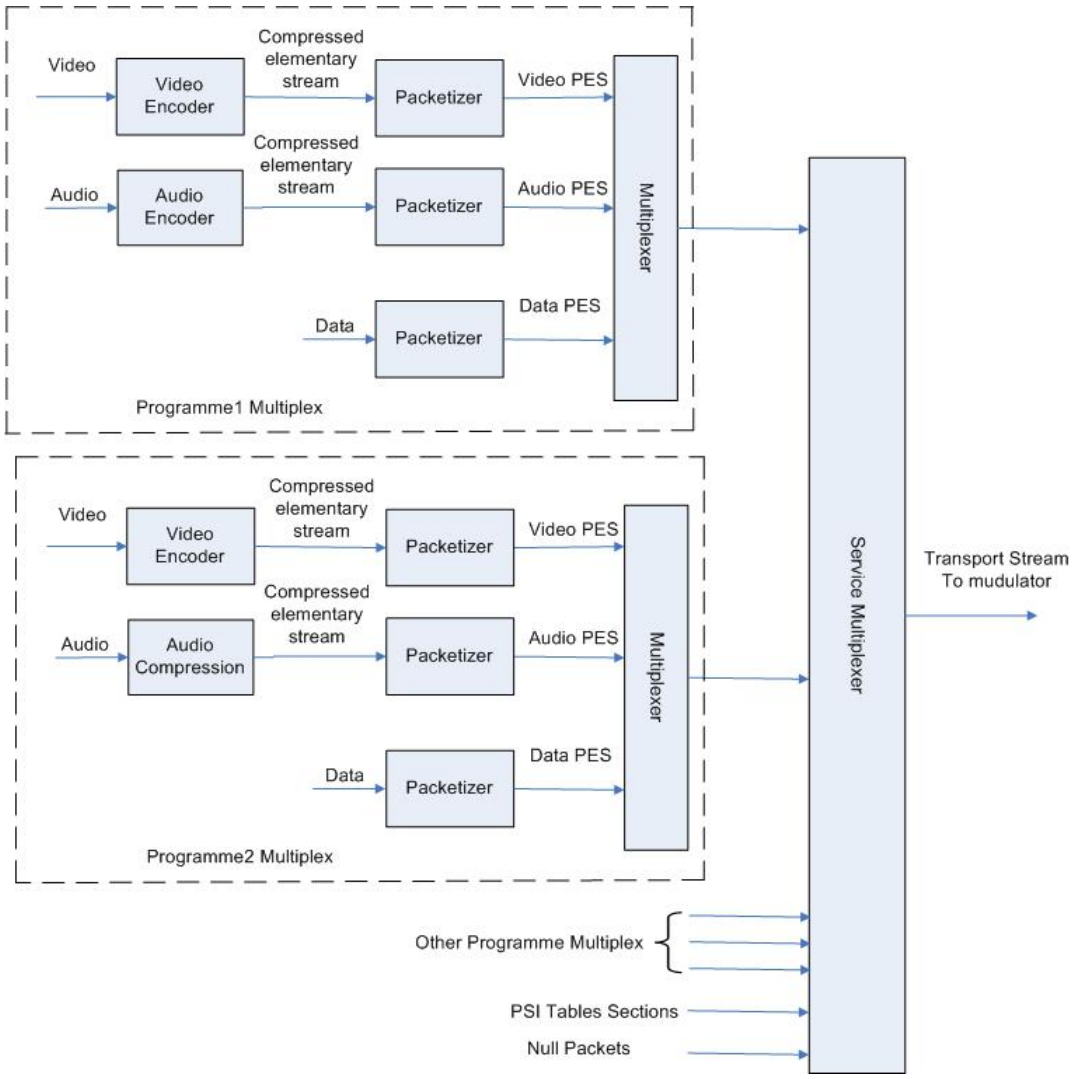

FIGURE 2.2: Multiple MPEG programs are combined into a transport streams

As showed in Figure 2.2, the overall multiplexing process can be described as a combination of multiplexing at two different layers. In the first layer, the program transport streams are formed by multiplexing one or more compressed elementary streams of the same programe. The compressed elementary stream (ES) is the output of an audio or video encoder consisting of a sequence of audio or video frames. The multiplexing process in the first layer can be divided into two steps:

First Divide each compressed elementary stream into a number of packetized elementary stream (PES) packets.

Second Combine all the packetized elementary streams of one programe to form a programe transport stream.

In the second layer multiple program transport streams as well as program-specific information (PSI) tables are combined to form a single transport stream to be transmitted. PSI talbes include some information that helps correctly demultiplexing of programs by decoders. More details about PSI are discussed in Section 2.2.4. Since some transmission schemes, such as those in DVB, impose strict constant bit-rate requirements on the transport stream, a multiplexer may need to insert some additional packets (called null packets) to maintain the physical bearer bit-rate when there is no sufficient TS packets to be sent. The payload of null packets may not contain any data at all, and the receiver is expected to ignore its contents.

2.2.2 Packetized elementary stream

A packetized elementary stream (PES) consists of a single compressed ES which has been made into packets, each starting with an added packet header. A PES stream contains only one type of data from one source, e.g. video or audio ES.

Each PES packet header contains a 24-bit start code prefix (0x000001) at the beginning, followed by an 8-bit stream ID, *PES_packet_length*, optional PES header fields and PES packet payload. For more detail about PES packet format, see Appendix B. The payload of a PES packet shall be contiguous bytes of data from the elementary stream indicated by the stream ID. Figure 2.3 shows the process of mapping the data in the compressed audio elementary stream to the payload of the packetized elementary stream.

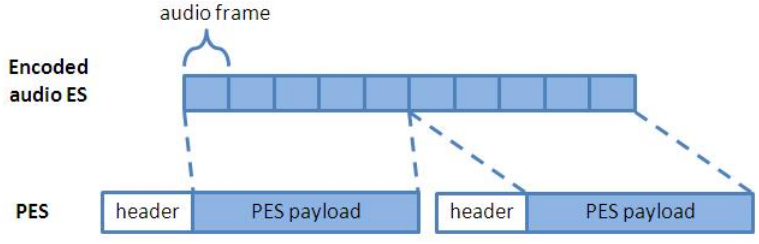

FIGURE 2.3: Elementary stream packetization

PES packets have variable length which is specified by the field *PES_packet_length* in the PES packet header:

$$PES\_packet\_Len = value\ of\ PES\_packet\_length + 6. \tag{2.2}$$

The length of a PES packet header, *PES_header_Len*, is also specified in the PES packet header:

$$PES\_header\_Len = value\ of\ PES\_header\_data\_length + 9 \tag{2.3}$$

Therefore, the length of the PES payload can be computed by the following equation:

$$\begin{aligned} PES\_payload\_Len &= PES\_packet\_Len - PES\_header\_Len \\ &= value\ of\ PES\_pacekt\_length - value\ of\ PES\_header\_data\_length - 3 \end{aligned} \tag{2.4}$$

2.2.3 Transport stream

TS packet format

A TS packet, 188 bytes in length, is the basic unit of data in a transport stream. The first 4 bytes of TS packet, named TS 4-byte prefix, consists of a sync byte (0x47), followed by three one-bit flags, a 13-bit Packet Identifier (PID), two 2-bit flags and a 4-bit continuity counter. The PID value is a 13-bit field, indicating the type of the data stored in the packet payload. Each PSI table or elementary stream in a transport stream is identified by a PID. The process of determining the PID of each elementary stream by PSI tables is described in section 2.2.4. The continuity counter increments with each transport stream packet with the same PID and it wraps around to 0 after its maximum value which is 15.

The optional adaptation fields may follow after the TS prefix which contains information used for timing and control. The presence of the adaptation field is indicated by adaption field control bits in TS prefix.

The rest of the packet consists of payload. The TS packet payload contains data from PESs or PSI tables. In case of null packets with PID 0x1FFF, the payload may be assigned any value. The number of data bytes in payload is specified by 184 minus the

number of bytes in the adaptation field. For more detail about transport stream format, see Appendix A.

Mapping PES data into TS payload

PES packets have variable length, not corresponding to the fixed packet length of transport packets, and may be much longer than a transport packet. Therefore, breaking up the data in PES packet into TS payloads is needed when multiplexing all the packetized elementary streams into a transport stream.

The mapping process is illustrated in Figure 2.4. The PES header is always placed at the beginning of a transport packet payload, immediately following the transport packet header, and shall not span multiple TS packets. The remaining PES packet content fills the payloads of successive transport packets until the end of the packet is reached. In addition, for the final transport packet, if there is insufficient PES packet data to completely fill the TS payload, stuffing bytes (0xff) in adaptation field are needed.

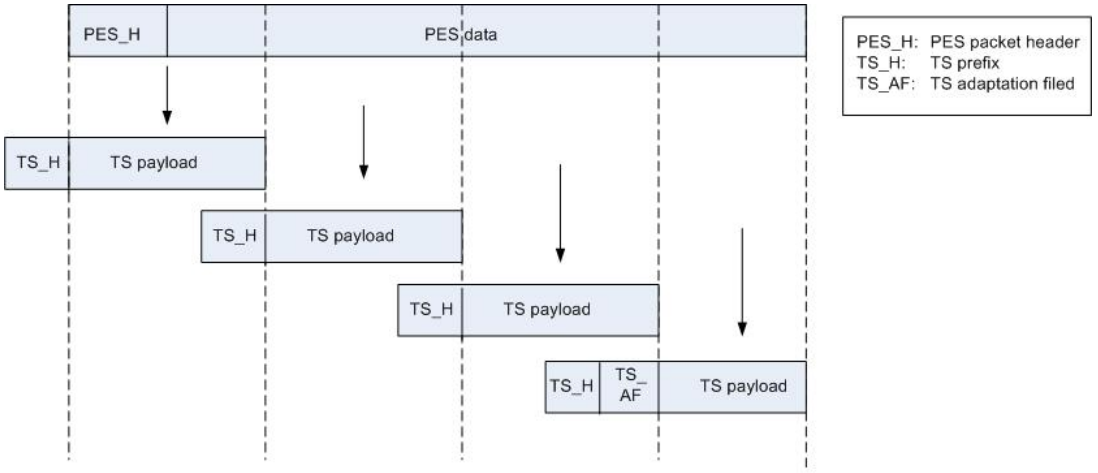

FIGURE 2.4: Mapping PES data into TS payload

Furthermore, if scrambling takes place when the data is inserted into TS packet, then the PES packet including its header is scrambled. Scrambling is indicated by *transport_scrambling_control* in the TS packet header.

2.2.4 PSI Tables & demultiplexing process

PSI tables carried in the transport stream describe the structure of that transport stream. The information contained in PSI is necessary to demultiplex a transport stream

and successful regenerate programs. Demultiplexing is a process that extracts elementary streams from the transport stream in part by looking for TS packets identified by the same PID. To identify the required PID, information in two PSI tables, Program Association Table (PAT) and Program Map Table (PMT), are needed. PMT specifies information about which PIDs, and therefore, which elementary streams are associated to form each program. PAT with PID value of 0x0000 provides the correspondence between a program number and the PID of PMT.

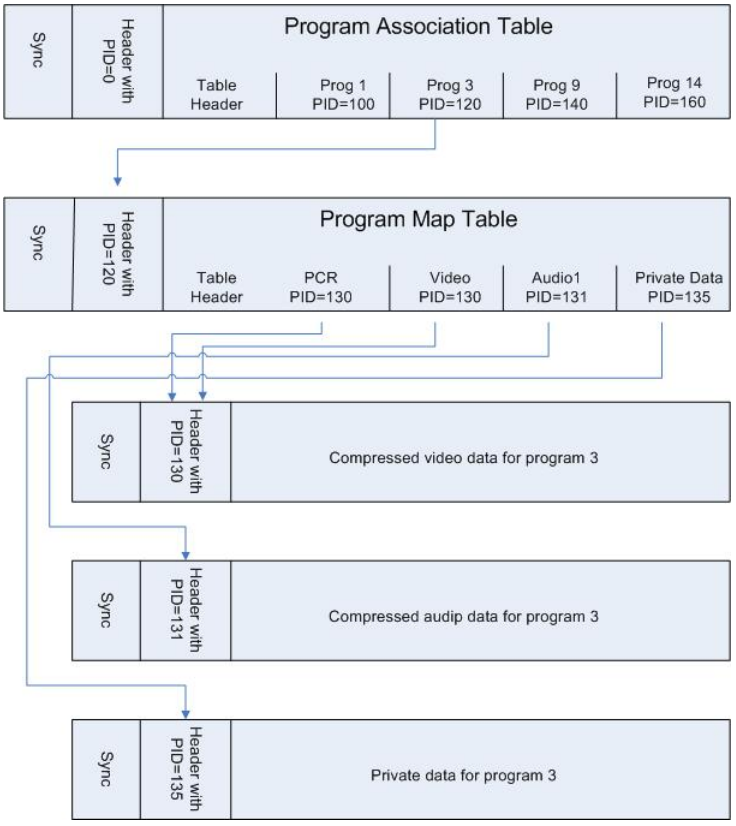

FIGURE 2.5: Process how required PID is determined [14]

The process of determining a PID is illustrated in Figure 2.5. In this example showed in Figure 2.5, a transport stream contains four programs: Program 1,3,9 and 14. The procedure of determining the elementary streams of Program 3 is as follows:

First Find TS packets with a PID of 0. The payloads of these packets compose the PAT.

Second According to PID information in PAT, find out the PID of PMT for Program 3.

Third According to PID information in PMT, find out the PID of all the streams in Program 3.

Chapter 3

Watermarking application solutions

This chapter introduces the digital watermarking algorithm used in the project and describes two watermarking application solutions that were developed in the project.

3.1 Digital watermarking algorithm

Before the embedding process, it is necessary to find a suitable watermarking algorithm for the project. Firstly, in order to effectively protect the multimedia content, the watermarking scheme should meet all the following requirements:

- The watermark should be perceivably transparent.
- The watermark should be robust enough to withstand both normal signal processing manipulations (e.g. lossy compression, A/D conversion, etc) and intentional removal attempts so that watermark can be reliably extracted at the detection side.
- The watermark should be secure so that unauthorized agents must not be able to detect the presence of embedded data.

Secondly, the watermarking algorithm should embed watermark data into MP2 audio data. As discussed in section 1.3, watermarks can be embedded into compresseed data or uncompressed data. Since audio data is stored and transmitted in MPEG-1 Layer II audio format in the project, compressed domain watermarking is more suitable in this scenario.

The watermarking scheme used in this project operates the MP2 audio data frame by frame. It periodically embeds the watermarks into a few samples in each audio frame. The watermarking procedure is firstly computes the changes of each sample to be watermarked and then adds or subtracts the changes based on the sample values.

3.2 Solution description

3.2.1 Server-guided watermarking

Considering the insufficient processing ability of receiving devices such as set-top box in the digital TV distribution system, [15] proposes a server-guided watermarking solution which is designed to be used in a CA/DRM environment. CA/DRM systems imposes secrecy on transmitted content by encrypting the data stream using a key that changes every fraction of a second up to every 20 seconds depending on the system [16]. The time between each key change is referred to as a *cypto-period*. This key is made only available to authorized user to decrypt the received data.

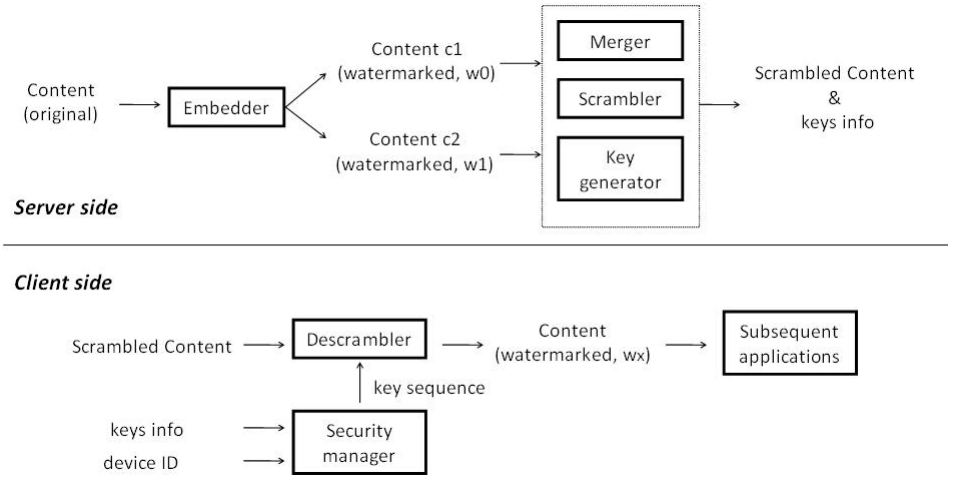

FIGURE 3.1: General system design diagram, [15]

Figure 3.1 demonstrates the general system designed for the server-guided watermarking solution mentioned above. At the server side, embedder creates two copies of the same audio stream and the two copies, c_1 and c_2, are watermarked by w_0 and w_1 respectively. Each watermark contains inter-independent symbols. In the merger, each watermarked copies is divided into segments. The length of each segment is same as that of a single watermark symbol. The segment from c_1 is scrambled with key k_1^s, while the segment from c_2 is encrypted with key k_2^s. s is the number of segment. The scrambled content with keys information is sent via DVB distribution network to the receiver. At the client side, the security manager extracts the key sequence from the received keys information, based on the receiver's identity information, such as client ID serial number in the smart card. The main principle of security manager is that it only makes choice between k_1^s and k_2^s for every segment s. The segment can only be correctly desrambled with a proper key. The segment descrambled by a wrong key can be easily filtered out by the subsequent application. Hence, a new watermark w_x can be generated. The process of generating a new watermark is illustrated in Figure 3.2.

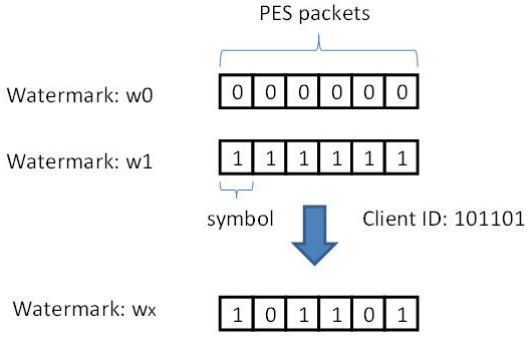

FIGURE 3.2: Selecting process at the client side

In Figure 3.2, one square represents a watermark symbol that is the minimal unit of the watermark. The length of a symbol should be longer than that of every PES packet. The square '0' and '1' respectively represent the symbol of watermark w_0 and w_1. A new watermark w_x can be generated depending on the client ID after the descrambling process.

The embedding algorithm proposed in this thesis is based on the watermarking solution mentioned above. The watermarking algorithm used in this solution is described in section 3.1, which embeds the watermark frame by frame. There are two approaches to manipulate the data and embed the watermark at the server: one is manipulating

the ES/PES before multiplexing the streams and the other is manipulating the TS after the service multiplexing. For the first method, two copies have the same stream ID and one copy would be discarded as the duplicated packets by the multiplexer. So, this one cannot work in the solution. However, if two copies are assigned two different stream ID, they would not be discarded in the multiplexer. The two watermarked copies are going to be demultiplexed and decoded as two different audio streams at the client side. In this case, an algorithm of switching between the two audio streams has to be designed and applied in the client device, which requires extra processing resources of the client device. While the second method does not need the switching process in the client device. It only needs extra data manipulation of reading the particular ES from the TS at the server side to embed the watermark. Because of the limited processing resources of the STB, the second method is chosen in the project. This algorithm manipulates the data of the transport stream at the headend to generate a new watermarked transport stream and it doesn't contain encryption operation. The basic structure of the headend is illustrated in Figure 3.3.

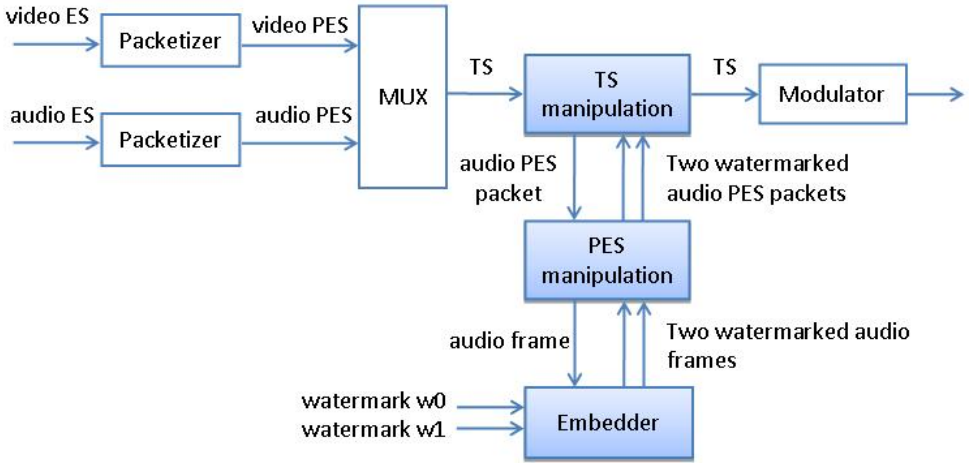

FIGURE 3.3: Basic structure of the headend for server-guided watermarking

At each time, TS manipulation module assembled an audio PES packet, which is about to be watermarked, from a transport stream. Then, an audio frame can be extracted from this PES packet through the PES manipulation module. In the embedder, two watermarks are embedded on the copies of the same audio frame respectively. The watermarking is done frame by frame. The outputs of the embedder are two differently watermarked audio frame copies. The watermarked data can be put back into the payload of the PES packet and then two watermarked PES packets can be got. The TS

manipulation module can insert these two watermarked PES packets back into the original transport stream. After manipulating all the audio PES packets to be watermarked, a new watermarked transport stream can be generated and later be transmitted out.

Moreover, besides the requirements discussed in section 3.1, the watermarking algorithm used in the embedder should also meet other requirements for this project:

- There is no dependency between the watermark symbols;
- The length of each symbol should be longer than one PES packet;
- The length of each symbol should be less than the *cypto-period* of the CA/DRM system. In this project, the *cypto-period* is 2 seconds.
- The watermarking algorithm should not change the length of the audio data.

3.2.2 Set-top box (STB) watermarking

The set-top box watermarking embeds the watermark at the set-top box (STB), so the STB must contain enough computational resources for a watermark embedder. Because of the small processing ability of STB, the complexity of the watermark embedding process gains importance. Low-complexity methods which preferably embed the watermark directly into the compressed audio bitstream are desirable. The watermarking algorithm can only operate the compressed data before the decoding process. Since the watermark is only embedded on the audio stream of one program that is going to be played out, the best choice is to manipulate the data of the audio PES after the demultiplexing process. Otherwise, if watermark is embedded before the demultiplexing, extra data manipulation of reading the PES from the TS has to be done at the client side in order to embed the watermark on the audio frame.

Figure 3.4 shows the structure in the client device for set-top box watermarking. A particular packetized elementary stream (PES) can be extracted by demultiplexing the received transport stream. After that, the PES manipulation module extracts the audio frame and later in the embedder the watermark are embedded into this audio frame. Then, the watermarked audio frames are put back into the original PES to generate a watermarked PES and this new PES is decoded to be played out.

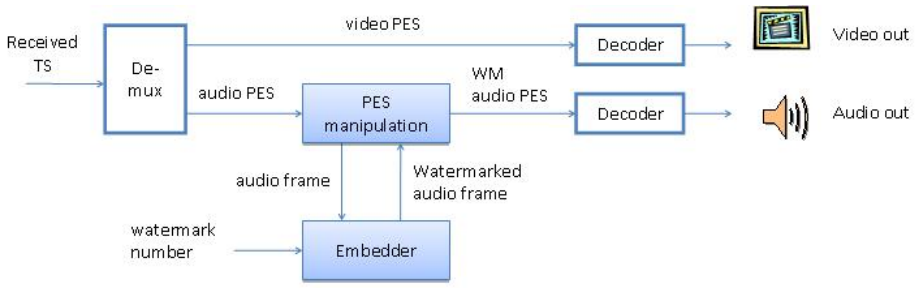

FIGURE 3.4: Basic structure in the client device for STB watermarking

Chapter 4

Implementation

The previous chapter introduces the basic idea and system architecture of two watermarking solutions. In this chapter, more detail things about the two solutions such as the feasibility study, algorithm, implementation and test are described.

4.1 Server-guided watermarking

The server-guided watermarking approach described in section 3.2.1 is chosen in this project and only the application at the server side is implemented.

4.1.1 Feasibility study

As described in section 3.2.1, the security manager can only choose to send out one key from k_1^s or k_2^s for every segment s. Thus, only one segment from either the first or second copy can be correctly descrambled with the right key and segments descrambled with a wrong key become 'rubbish' content. However, it is possible that the 'rubbish' content can cause audible artifacts when playing the audio content. The purpose of the feasibility study is to check whether the data manipulations in the project would cause a perceptible audio artifact when the watermarked audio content is played.

In order to exclude the possibility that the artifacts are caused by the watermarking algorithm, the application in this study doesn't contain the watermark embedding process. A simple bitwise logical operator, exclusive disjunction (XOR), is used instead to

simulate the descrambling with a wrong key. XOR manipulates the sequence of bits. In fact, any logical operator can be used instead of XOR if it can change the bits of the PES packet header, so that this PES packet can be filtered out by the subsequent application. Besides, the embedding algorithm is only applied on the audio stream from one program. All the transport stream (TS) packets containing the audio data of this program can be extracted from the transport stream in terms of their PID value. Moreover, because every TS packet can only store data from one PES, the operation can be done by reading PES packet by PES packet.

Algorithm

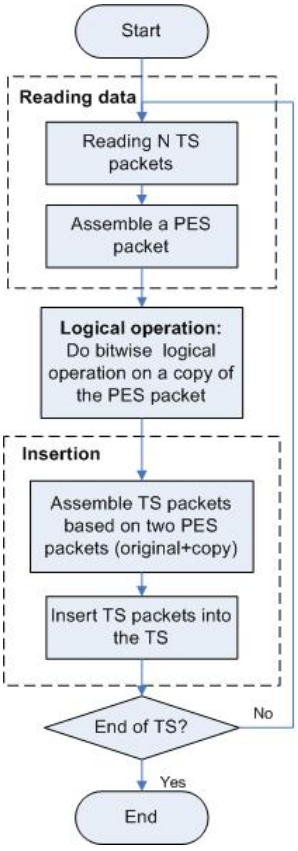

FIGURE 4.1: Algorithm designed for feasibility study

The algorithm for this feasibility study is indicated in Figure 4.1. The main loop in this algorithm contains three steps (Reading data, Logical operation and Insertion) which are repeatedly executed until the end of the transport stream is reached. Figure 4.2 shows the data implementation on the transport stream. In this figure, the sequence of squares represents part of a transport stream which contains data from multiple programs and

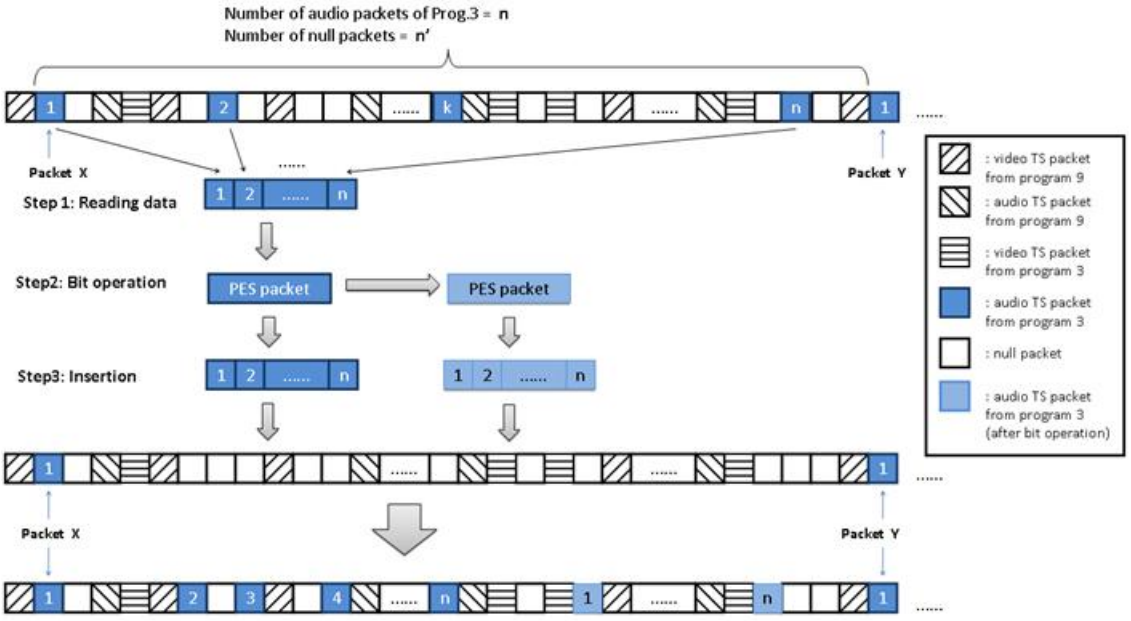

FIGURE 4.2: Algorithm operating on a TS

each square represents a TS packet. Different colors and textures of the square indicate different types of data in the TS packet, which is showed in Figure 4.2. In this example, only audio TS packets containing audio data from Program 3 (blue squares) are going to be manipulated. The following three paragraphs go into details of the three steps.

Step1: Reading data

This step includes two sub-steps. The first one is to read n TS packets from the transport stream. In the example illustrated in Figure 4.1, the TS packets whose PID values identifying the audio PES of Program3 are extracted. All the payloads of the n audio TS packets comprise one audio PES packet from a particular PES (in this example, one audio PES of Program 3). The first TS packet of the PES packet, denoted by Packet X in Figure 4.2, can be found when reading the payload that contains the 24-bit start code prefix (0x000001) of the PES packet. The TS packet denoted by Packet Y contains the start code prefix of the next PES packet. The number of n can be easily counted after extracting all the TS packets with matching PID value between Packet X and Packet Y, including Packet X. Besides, The number of null packets between Packet X and Y is $n^{'}$. The second sub-step is to assemble a PES packet from the payloads of the n TS packets, which is the reverse process of mapping the PES data into TS payloads described in section 2.2.3.

Step2: Logical operation

This step makes a copy of the PES packet assembled in Step 1 and performs bitwise logical operation XOR (exclusive OR) on that copy.

Step3: Insertion

This step contains two sub-steps. The first sub-step is to creates n new TS packets. The header and possibly adaptation part of the n new TS packets are same as that of the original ones in Step1. The payloads of these new packets are contiguous bytes from the PES packets got in Step2. In the second sub-step, all the original TS packets in the trasport stream are replaced with null packets except the first TS packet, Packet X, and then $n-1$ original TS packets (excluding Packet X) and n new TS packets are inserted into the transport stream. Insertion is a process of replacing some of the null packets, which is allocated between Packet X and Y, with the $2n-1$ TS packets. In order to evenly insert the TS packets, a counter is created which increments with each null packet after Packet X and the $\lfloor \frac{n+n^{'}}{2n-1} \rfloor$-th null packet are replaced by a TS packet. The counter wraps around to 0 after its maximum value $\lfloor \frac{n+n^{'}}{2n-1} \rfloor$. The insertion order for the $2n-1$ TS packet is that first insert $n-1$ original TS packets and then insert n new TS packets. For $n-1$ original TS packets, they should be inserted according to their original position in the transport stream. The TS packet which was located at the beginning should be inserted first. The n new TS packets are also inserted in the same way. At last, a new transport stream is generated. In the example, $\lfloor \frac{n+n^{'}}{2n-1} \rfloor$ is equal to 2, so a TS packet replaces the null packet each time when counter reaches 2. Moreover, The number of null packet, $n^{'}$, is larger than that of the new TS packets, n. The $n-1$ original TS packets from the 2nd one to the n-th one in the original TS and the n new TS packets are inserted into the TS.

After manipulating the first PES packet, the embedding algorithm repeats the above three steps to continually manipulate all the remaining audio PES packets from the paricular PES until the end of the transporst stream is reached.

Implementation & Test

The feasibility study is carried on the PC. The transport stream used in the project is a .ts file with 262,144 Kbytes, containing data from seven TV programmes. The length of the audio PES to be manipulated is about 1.5 minutes. The audio data is compressed

according to MPEG-1 Layer II (MP2). The sampling frequency of this audio stream is 48 KHz, and the bit-rate is 192 Kbps (Joint Stereo mode). Therefore, according to the equation 2.1 in section 2.1, the length of each audio frame in the example transport stream is 576 bytes.

The set-up for the feasibility study is showed in Figure 4.3.

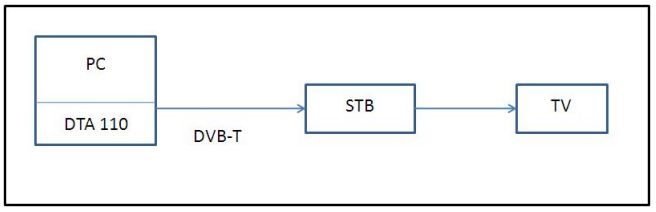

FIGURE 4.3: Test set-up

DTA 110, a DVB-T modulator with UHF up-converter, is connected to a personal computer (PC) by PCI (Peripheral Component Interconnect) bus. RF output of PC is connected directly to a set-top box (STB) by a cable and the STB is connected to a TV set. The transport stream can be transmitted out by the DTA-110. The set-top box receives the signal, turns the signal into program content and displays the content on the TV.

The last thing is testing the perceptual quality of the audio content of Program3. There are two possible audio artifacts caused by the data manipulation:

Click: Decoder tries to play the 'rubbish' packets. Here, the rubbish packets are the PES packets operated by XOR.

Playing speed change: Decoder estimates the bitrate considering the number of 'rubbish' packets.

In the project, none of these artifacts can be heard, so the data manipulation in this study is feasible.

4.1.2 Watermarking application

The embedding algorithm integrates the watermarking algorithm with other data manipulation applications as showed in Figure 3.1. TS manipulation module assembles one

PES packet and puts two PES packets back into the transport stream, using the same operations as described in section 4.1.1 except the bitwise logical operation. Generally speaking, the embedding algorithm reads the audio data that is about to be watermarked, does watermarking twice on it and inserts the watermarked data back into the transport stream.

Algorithm

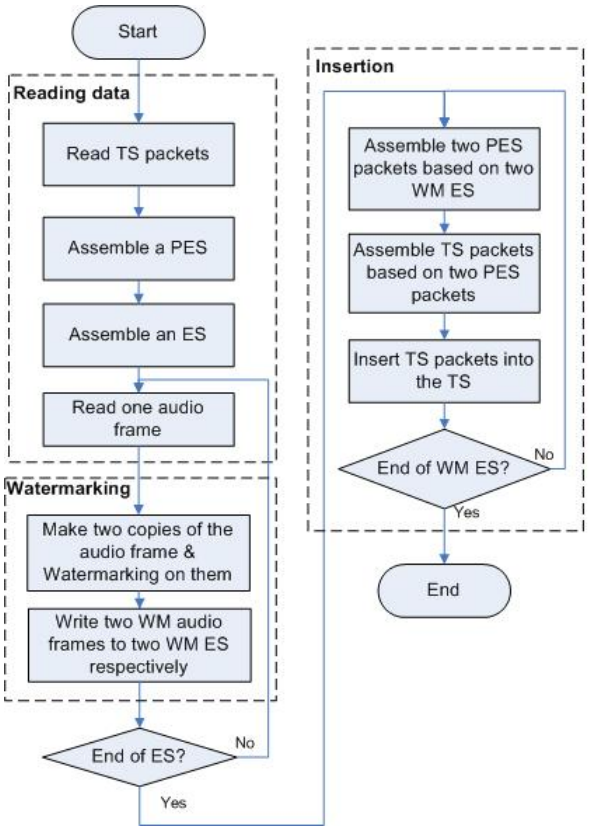

FIGURE 4.4: Embedding algorithm

The embedding algorithm for the watermarking application at the server side is indicated in Figure 4.4. This algorithm contains three main steps: Reading data, Watermarking and Insertion. Similar to Figure 4.2 in section 4.1.1, Figure 4.5 shows the data manipulation process of embedding algorithm on the transport stream.

Step1: Reading data

This step includes four sub-steps in which the first two sub-steps are similar to Step1 in section 4.1.1. For the first sub-step, the difference between the two algorithm is that here all the TS packets containing data from an audio PES of a particular program

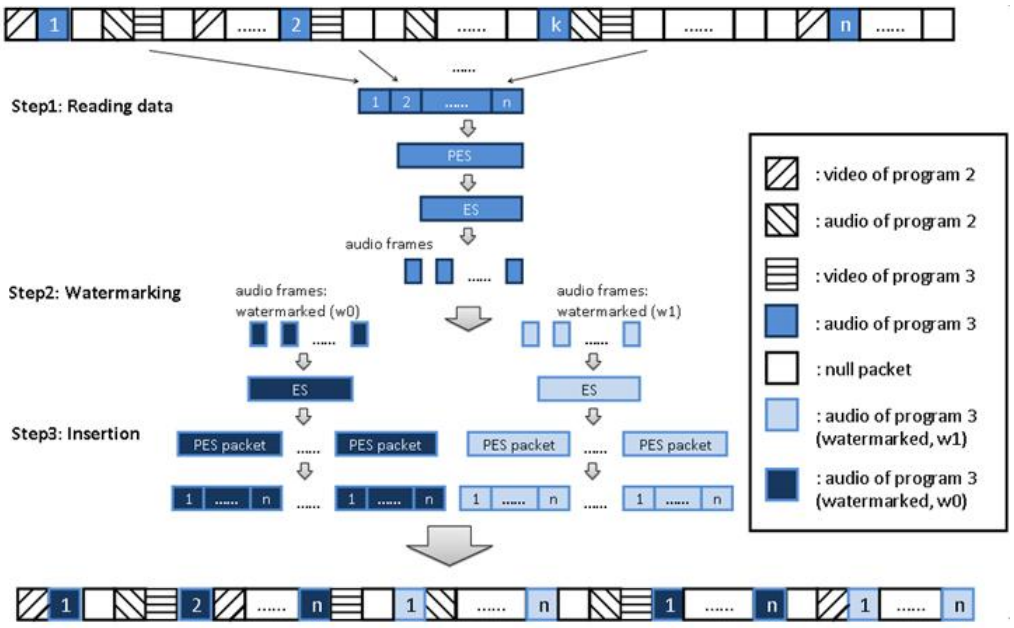

FIGURE 4.5: Data manipulation of embedding algorithm

(Program 3) are read from the transport stream instead of reading only n TS packets in the previous algorithm. For the second sub-step, the difference is that this algorithm assembles a whole packetized elementary stream based on the TS packets extracted in the first sub-step, while in the previous algorithm, it only assembles one PES packet at each time. Since the watermarking algorithm used in the project can only operate frame by frame, in the third sub-step a whole elementary stream is assembled based on the PES got in the second sub-step and in the last sub-step an audio frame is read from the ES. Every audio frame begins with a 12-bit sync word, 0xFFF, and its length is constant which is indicated in the frame header. The way to compute the length of the audio frame is discussed in equation 2.1.

Step2: Watermarking

This step first makes two copies of the audio frame that is read in Step1. Then, it embeds watermark w_0 on the first copy and watermark w_1 on the second copy. After that the two watermarked audio frames are written to two new elementary streams respectively. Therefore, after all the audio frames are written out, two differently watermarked copies of the same ES are generated. The watermarking algorithm used in this step is described in section 3.1.

Step3: Insertion

Insertion part has a loop including three sub-steps. The first sub-step assembles two

watermarked PES packets. In this sub-step, the i-th ($i = 1$, at first) PES packet of the PES in the second sub-step of the Extraction part is read. This sub-step also makes two copies of this PES packet and replaces the payloads of the two copies with contiguous bytes of data from two newly generated elementary streams in Step2 respectively. The PES payload filling process is showed in Figure 2.3. Thus, two watermarked PES packets are generated. Because the client device like STB cannot play two audio streams at the same time, one solution is to filter out one audio stream by the decoder, which can be done by changing some bits of the header in the second watermarked PES packet. Furthermore, the data manipulation process of the last two sub-steps is similar to that of the Step3 in the feasibility study. In the second sub-step, two groups of n TS packets are assembled based on the two watermarked PES packets and the n original TS packets. The TS packets assembling process is same as the first sub-step of Step3 in the feasibility study. n depends on the length of the PES packet. Data in the PES packet can be mapped into the payloads of n TS packets. At last, the n original TS packets in the transport stream are replaced by null packets and $2n$ TS packets are inserted into the transport stream in the same way of second sub-step of Step3 in the feasibility study. Assume that the number of null packets is larger than the number of the TS packets of one copy. After executing all the three sub-steps, the algorithm immediately checks if all the data in the watermarked elementary stream is operated. If not, the remaining bytes in the ES should be used to assemble another two PES packets based on the $i+1$-th PES packet of the PES assembled in Extraction part. Otherwise, the algorithm is finished. A new transport stream with two differently watermarked copies of the same data is generated.

Implementation & Test

This watermarking application is carried on the PC. The transport stream and test set-up used in the project is exactly the same as that of the feasibility study described in section 4.1.1. The difference is, here we not only test the perceptual quality of the audio content, but also try to detect the embedded watermarks.

1. Perceptual Quality

The test method is mentioned in section 4.1.1. Acoustical inspection of watermarked sequences both on computer and on TV sets did not yield any perceptible

audio artefacts. In another words, the watermarked audio signal played on TV cannot be perceptually discriminated from the original signal due to embedding process.

2. Watermark detection

We assume the decoded bit-stream can be captured before playout on the device. This scenario is simulated by saving one watermarked audio elementary stream generated in Step 2 into a file. On the computer, the watermark can be detected by the watermark detector and then be compared with the original watermark. In the project, the detected watermark and the original one are the same. Therefore, the embedding algorithm described in this thesis can manipulate the data to correctly embed the watermark at the server side.

4.2 Set-top box (STB) watermarking

The solution focuses on the design of PES data manipulation module. Each time a block of data is read from the given PES and push to a buffer for further operation. Considering the small processing resources of the STB, the watermarking solution for STB is different from the server-guided solution. Once the watermark is embedded into the audio frame, the watermarked frame should immediately be written out. The algorithm is described below.

4.2.1 Basic algorithm design

For simplicity, assume that the size of the buffer (N bytes) created to put the input data is much larger than that of an audio frame, so that watermarking algorithm can directly operate on a complete audio frame. The algorithm flow chart is showed in Figure 4.6.

Before entering the main loop, the algorithm first read data from the input PES stream and push it into the buffer which has a size of N bytes. Then it writes the data which is allocated before the first frame sync word (0xFFF) to the output stream.

1.Inner loop

There are three steps in the inner loop:

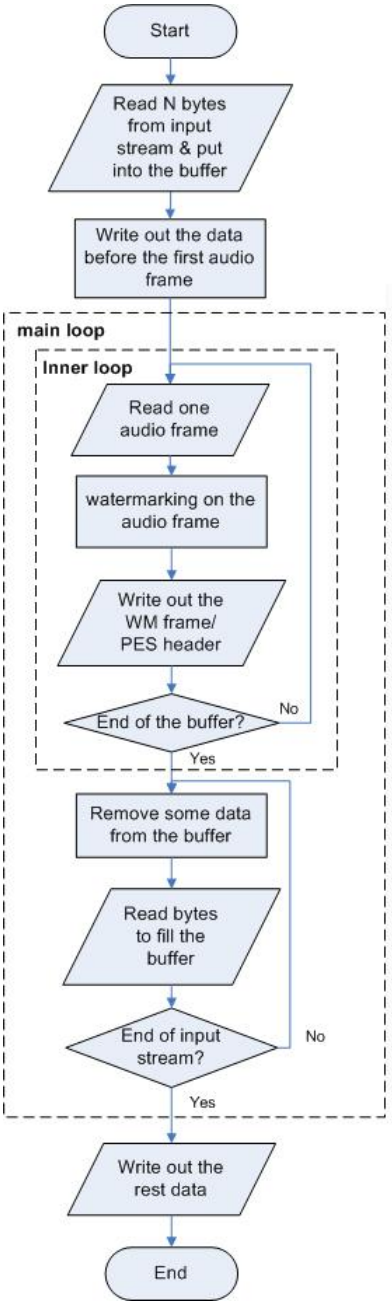

FIGURE 4.6: Algorithm for STB watermarking solution

- Read one audio frame whose length is constant and specified in its header as described in section 2.1.
- Embed the watermark on the audio frame.
- Write the data from the watermarked audio frame or the PES packet header to the output stream. If the audio frame is distributed in two PES packets, the data should be written out in this order: part of the watermarked audio frame in the first PES packet, the header of the second PES packet and part of the watermarked frame in the second PES packet. Otherwise, if all the data of the audio frame is in one PES packet, the whole watermarked audio frame should be directly written to the output stream.

All these three steps are repeatedly executed until the end of the buffer is reached.

2.Main loop

If the last audio frame read in the buffer is incomplete, keep the data of this frame in the buffer and remove all the data before this frame. Next, read more data from the input stream to fill the buffer and execute the steps in the inner loop until the end of the input stream is reached.

The last thing is to write all the remaining data in the buffer to the output stream. Therefore, the output stream contains a watermarked PES.

Implementation & Test

In this project, the set-top box watermarking application is simulated by PC. The input stream is an audio PES which is directly extracted from the transport stream file and begins with the sync word, 0x000001. The buffer size is $N(N = 8192)$ bytes. The size of one audio frame in the PES example is 576 bytes. In order to detect the watermark, an elementary stream is extracted from the output stream. For the detail of the extraction process, refer to the algorithm in section 4.1.2. In the project, the correct watermark can be detected from the elementary stream.

4.2.2 Development

Because of the comparatively small processing ability of set-top box, the design in section 4.2.1 is impractical for most set-top boxes. In reality, the set-top box can only read a small amount of data to operate for each time. Assume that the set-top box read $N^{'}$ ($N^{'}$ = 150bytes) of data, called an arrival data block, from the input stream at a time. As described in section 3.1, the watermark is inserted only to a few samples in each frame, so other data in the same frame will not be changed. Since the reading of sample part depends on information from frame header, allocation section and SCFSI section as discussed in section 2.1, values in these sections should be saved separately in case the arrival data block only contains partial audio frame. Furthermore, once finishing the data manipulation on one arrival data block, the set-top box immediately writes that data block out. It is impossible to manipulate two data blocks at the same time. So, if data of a sample to be watermarked is allocated in two data block, the watermarking cannot be performed. Because of the error tolerance of the watermark detector, a reasonable choice is to skip a sample allocated in different data blocks.

The algorithm flow diagram is showed in Figure 4.7.

The detail procedure in the main loop is as follows:

1. Read $N^{'}$ bytes of data from the input stream.
2. If the arrival data block contains data from frame header, allocation section or SCFSI section which is called information data, values of fields from these three sections shall be saved separately.
3. If the arrival data block contains a complete sub-band sample to be watermarked, embed watermark on the sample.
4. Write the whole data block to the output stream.

At last, repeat four steps in the main loop until all the data is written out.

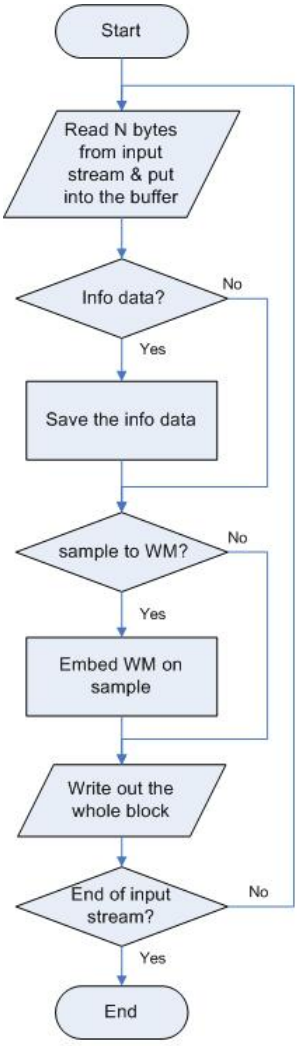

FIGURE 4.7: Algorithm for developed STB watermarking solution

Chapter 5

Conclusion

Forensic watermarking is a multimedia content protection technology that allows tracking back illegal copies of the protected multimedia content. In this thesis, two watermark embedding solutions for DVB are described, including their design and implementation. One is the server-guided watermarking and the other is the set-top box watermarking. Both of them enable embedding of a unique watermark to multimedia content for each client device. Both solutions have their own advantages and disadvantages and can be compared in the following three respects:

- *Processing resource usage of client device*

  The server-guided watermarking embeds watermarks at the server and only does some low-complexity operations at the client side, such as generating a key sequence and descrambling the watermarked content, which can be done fast on the hardware components of the client device, so only a little extra resources are required for the client device. While the STB watermarking embeds watermarks at the STB, which is a computationally heavy process and hence requires more processing resources to execute. The complexity of the watermarking algorithm gains more importance.

- *Audio quality*

  For server-guided solution, the embedding algorithm performed at the server requires sufficient null packets contained in the transport stream to correctly insert the watermarked audio copies. The number of null packet in the transport stream should be equal or higher than that of the audio TS packets to be watermarked.

In the project, the requirement is met. However, if null packets are not sufficient, in order to meet the requirement, the system has to increase the number of null packets in the transport stream by decreasing the bitrate of the audio stream, which can cause the drop of the audio quality. While for STB watermarking, its embedding algorithm does not have any influence on the audio quality.

- *Exposure of watermarking algorithm*
  Since the watermarking algorithm in server-guided solution embeds watermarks at the server side, there is no potential danger to expose the watermarking algorithm to the hacker. However, the watermarking algorithm in the STB watermarking solution may be exposed which renders the watermark vulnerable to counter measures.

Moreover, the watermarking algorithm used in this project is a compressed domain audio watermarking method. Since this algorithm can only operate with MPEG Layer II audio format, if the audio data is compressed with different audio format, the algorithm should be changed accordingly. The two solutions can also use compressed domain video watermarking instead of audio watermarking. However, the server-guided watermarking solution requires considerable useful bandwidth for broadcasting the watermarked copies of video data, which may cause a significant drop of the audio stream quality. For STB watermarking solution, the feasibility of video watermarking depends on the complexity of the watermarking algorithm and the computational resources of the STB.

In the project, the embedding algorithms for two solutions described in this thesis are implemented on the PC and both of them can correctly manipulate the data to embed the watermark on the audio stream of one program. The embedded watermarks can later be detected from the elementary stream. In reality, the embedding algorithm of the server-guided solution can operate on the audio stream of multiple programs as well. In server-guided solution, the process of scrambling and descrambling with the wrong key is simulated by changing the bits of the PES packet header, so that this PES packet will be filtered out by the decoder. Moreover, according to the test, the embedding process does not cause any perceptible audio artifacts.

In the future, for the server-guided watermarking, the whole system could be implemented which includes scrambling the watermarked data at the server side and descrambling as well as generating the key sequence at the client side. Furthermore, for

the STB watermarking, the implementation on the set-top box should be further done. The hardware debugging set-up is described in Appendix D.

Appendix A

MPEG audio frame header format

Table A.1: MPEG-1 audio frame header

Name	Number of bits	Description
syncword	12	Bit string '1111 1111 1111'
ID	1	'1' for MPEG audio, '0' is reserved.
layer	2	'11'= Layer I; '10'= Layer II; '01'= Layer III; '00'= reserved.
protection bit	1	'1' means no redundancy has added for error detection and concealment.
bit rate index	4	(See Table A.2)
sampling frequency	2	'00'= 44.1 kHz; '01'= 48 kHz; '10'= 32 kHz; '11'= reserved.
padding bit	1	'1' means presence of padding bit
private bit	1	'1' means private use
mode	2	00 - Stereo 01 - Joint Stereo (Stereo) 10 - Dual channel (Two mono channels) 11 - Single channel (Mono)
mode extension	2	(Only used in Joint Stereo) 00 - bands 4 to 31 01 - bands 8 to 31 10 - bands 12 to 31 11 - bands 16 to 31
copyright	1	'1' means copyright
original	1	'1' means the bitstream is an original; '0' means it is a copy.
emphasis	2	00 - none 01 - 50/15 ms 10 - reserved 11 - CCIT J.17

TABLE A.2: Bit rate (in kilobits per seconds)

bit_rate_index	**bit rate (Kbit/s)**		
	Layer I	Layer II	Layer III
0000	Free format		
0001	32	32	32
0010	64	48	40
0011	96	56	48
0100	128	64	56
0101	160	80	64
0110	192	96	80
0111	224	112	96
1000	256	128	112
1001	288	160	128
1010	320	192	160
1011	352	224	192
1100	384	256	224
1101	416	320	256
1110	448	384	320
1111	Reserved		

Appendix B

Packetized elementary stream format

Table B.1: PES packet format

Name	Size	Description
Packet start code prefix	3	bytes 0x000001
Stream id	1	byte Examples: Audio streams (0xC0-0xDF), Video streams (0xE0-0xEF)
		Note: The above 4 bytes is called the 32 bit start code.
PES Packet length	2	bytes Can be zero as in not specified for video streams in transport streams
Optional PES header	variable length	(see Table B.2)
		Note: all the fields above is called PES header.
PES packet data byte		See elementary stream. In the case of private streams the first byte of the payload is the sub-stream number.

TABLE B.2: Optional PES header

Name	Number of bits	Description
marker bits	2	10 binary or 0x2 hex
PES scrambling control	2	'00' = Not scrambled. '01' = Reserved for future use, '10' = Scrambled with even key, '11' = Scrambled with odd key
PES priority	1	1 indicates a higher priority of the payload of the PES packet.
data alignment indicator	1	1 indicates that the PES packet header is immediately followed by the video start code or audio sync word.
copyright	1	1 implies copyrighted.
original or copy	1	1 implies original. 0 implies copy.
PTS DTS flag	2	'10'= PTS shall be present; '11'= both PTS and DTS shall be present; '00'= no PTS or DTS shall be present; '01' is forbidden
ESCR flag	1	1 implies ESCR and extension fields are present in PES packet header (Not need to be coded in DVB system).
ES rate flag	1	1 implies ES_rate is present in the PES packet header (Not need to be coded in DVB system)
DSM trick mode flag	1	1 implies the presence of an 8-bit trick mode field
additional copy info flag	1	1 implies the presence of the additional_copy_info
PES CRC flag	1	1 implies the presence of a CRC in the PES packet
PES extension flag	1	1 implies the presence of an extension field in PES packet header
PES header data length	8	gives the length of the remainder of the PES header
Optional fields	variable length	presence is determined by flag bits above
Stuffing bytes	variable length	0xff

Appendix C

Transport stream format

[1] $N = 184 -$ number of bytes in *adaptation_field*

TABLE C.1: Transport stream packet format

Name	Number of bits	Description
sync_byte	8	0100 0111 or 0x47
transport_error_indicator	1	Set by demodulator if at least one uncorrectable bit error exists in the associated transport stream packet
payload_unit_start_indicator	1	1 means start of PES data or the first byte of PSI otherwise zero only.
transport_priority	1	One means higher priority than other packets with the same PID.
PID	13	Packet ID
transport_scrambling_control	2	'00' = Not scrambled '01' = Reserved for future use '10' = Scrambled with even key '11' = Scrambled with odd key
adaptation_field_control	2	'00' = Reserved for future use; '01'= payload only; '10' = adaptation field only; '11'= adaptation field followed by payload
continuity_counter	4	Increase by 1 each time a source payload is present in the payload field and it wrap to '0000' from '1111'.
		Note: the total number of bits above is 32 and is called the transport stream 4-byte prefix.
adaptation_field	0 or more	Depends on flags(see Table C.2)
payload_data	$8 * N$[1]	Depends on flags. Payload can be video, audio, data, ECM or EMM, etc.

TABLE C.2: Partial adaptation field format

Name	Number of bits	Description
adaptation_field_length	8	Number of bytes in the adaptation field immediately following this byte. For inserting a single stuffing byte in a Transport Stream packet carrying PES packets, the value equals '0000 0000'.
discontinuity_indicator	1	Set to 1 if a discontinuity occurred in the continuity counter or system time-base.
random_access_indicator	1	Set to 1 if the PES packet in this TS packet starts a video/audio sequence
elementary_stream_priority_indicator	1	1 = higher priority
PCR_flag	1	1 means adaptation field does contain a PCR field.
OPCR_flag	1	1 means adaptation field does contain a OPCR field.
splicing_point_flag	1	1 means presence of splice countdown field in adaptation field.
transport_private_data_flag	1	1 means presence of private data bytes in adaptation field.
adaptation_field_extension_flag	1	1 means presence of adaptation field extension.
Below fields are optional	variable	Depends on flags
PCR	$33 + 9$	Program clock reference
OPCR	$33 + 9$	Original Program clock reference. Helps when one TS is copied into another
splice_countdown	8	Indicates how many TS packets from this one a splicing point occurs (may be negative)
stuffing_bytes	variable	8-bit value '1111 1111'

Appendix D

Debugging set-up for STB watermarking

The set-up for testing is illustrated in the figure below.

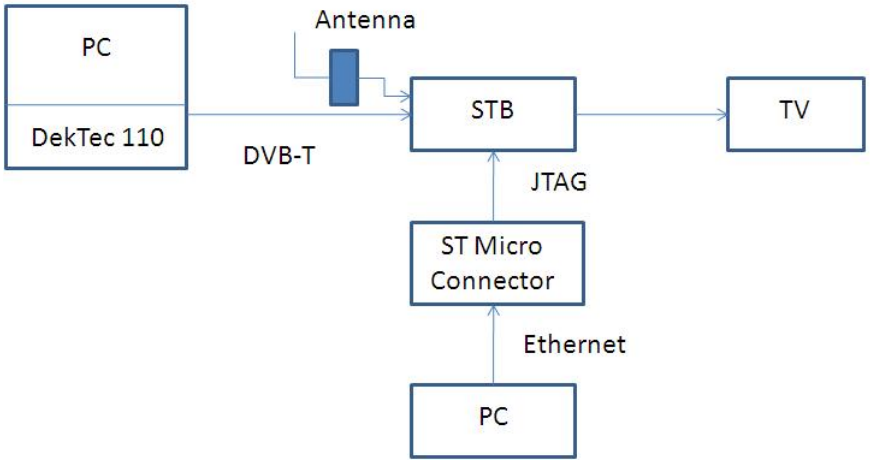

FIGURE D.1: Set-up for hardware debugging

The ST Micro Connect is connected to the host PC with an Ethernet network connection and to the target, set-top box (STB), with the JTAG port. The IP address of the ST Micro-connector is configured using the serial port. The IP address is added as a named target in the targets.cfg:

target st_eth tap "jei_soc IP address tckdiv=2" board_runtime_init

The ST Micro Connector is an emulator supporting the hardware debugging. It can emulate the programme running on the set-top box (STB).

All the codes should be rewritten to C language which can be compiled on the emulator. Since the emulator has its own library, all the code and use the library in emulator to receive and send the data.

Bibliography

[1] wikipedia, 2009. URL http://en.wikipedia.org/wiki/Cryptography.

[2] Nedeljko Cvejic. Algorithms for audio watermarking and steganography, 2004.

[3] wikipedia, 2009. URL http://en.wikipedia.org/wiki/Dvb.

[4] ETSI TR 101 154 v1.4.1. *Digital Video Broadcasting (DVB); Implementation Guidelines for the Use of MPEG-2 Systems, Video and Audio in Satellite, Cable and Terrestrial Broadcasting Applications*, 2000.

[5] *ISO/IEC 11172-3: Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s - Part 3: Audio*, 1993.

[6] *ISO/IEC 13818-3: Information technology- Generic coding of moving pictures and associated audio: Audio*, November 1994.

[7] *ISO/IEC 13818-7: Information technology- Generic coding of moving pictures and associated audio information- Part 7: Advanced Audio Coding (AAC)*, February 2006.

[8] ITU-R BS.1196-E. Audio coding for digital terrestrial television broadcasting, 1995.

[9] *ISO/IEC 13818-1: Information technology - Generic coding of moving pictures and associated audio information: Systems*, December 2000.

[10] Digital Watermarking Alliance, 2007. URL http://broadcastengineering.com/products/widevines-mensor-watermarking-system/.

[11] Jim Barratt. Classification in the digital age: a scoping study of future options, 2007.

[12] M.Steinebach, E.Hauer, and P.Wolf. Efficient watermarking strategies. In *Third International Conference on Automated Production of Cross Media Content for Multi-channel Distribution*, 2007.

[13] N.Cvejic and T.Seppanen. *Digital audio watermarking techniques and technologies: applications and benchmarks.* Information Science Reference, 2008.

[14] BBC Training & Development. Bbc training & development: Digital video broadcasting, 1999.

[15] D.Jarnikov and J.G. Lourens. Server-guided watermarking for resource-constrained devices. Unpublished manuscript, 2009.

[16] E.P.J.Tozer. *Broadcast engineer's reference book.* Elsevier, 2004.