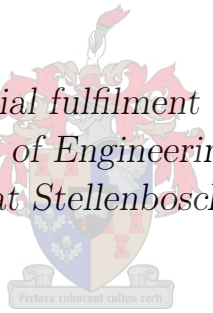
The Design and Implementation of a Prototype Digital Video Watermarking Scheme with Dynamic Traitor Tracing

by

Antony Salotto



Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering in the Faculty of Engineering at Stellenbosch University

Department of Electrical and Electronic Engineering,
Stellenbosch University,
Private Bag X1, Matieland 7602, South Africa.

Supervisor: Prof. G-J. van Rooyen

December 2014

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 2014 .....

Abstract

The Design and Implementation of a Prototype Digital Video Watermarking Scheme with Dynamic Traitor Tracing

A. Salotto

Department of Electrical and Electronic Engineering,
Stellenbosch University,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MEng (Electr and Electron)

December 2014

We design and implement a prototype digital watermarking scheme that uses spread-transform dither modulation to embed hidden binary message signals into uncompressed digital video signals. The hidden messages are intended to identify the original recipient of a particular copy of a digital video, so that in the event that a pirate copy of that video is found, the recipient can be incriminated as a pirate. We test the ability of the watermark signal to survive various modifications to the watermarked video, including lossy compression, frame deletion, and amplitude scaling. We find that the scheme performs well when watermarked videos remain uncompressed, but the scheme is not sufficiently robust against lossy compression. We also implement a dynamic traitor tracing scheme based on the scheme of Fiat and Tassa, which is designed to identify a group of pirates who combine their watermarked videos in an attempt to defeat the watermarking scheme. We test the traitor tracing scheme in a simulated digitally watermarked video broadcast system where a pirate rebroadcast is monitored. We find that when the pirate rebroadcast is delayed, our traitor tracing scheme identifies and disconnects the pirates more quickly than the scheme of Fiat and Tassa.

Uittreksel

Die Ontwerp en Implementering van 'n Digitale Videowatermerkskema-prototipe met Dinamiese Verraaier-opsporing

("The Design and Implementation of a Prototype Digital Video Watermarking Scheme with Dynamic Traitor Tracing")

A. Salotto

Departement Elektriese en Elektroniese Ingenieurswese,
Stellenbosch Universiteit,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MIng (Elektr en Elektron)

Desember 2014

Ons ontwerp en implementeer 'n digitale watermerkskema-prototipe wat strektransform-bibbermodulasie gebruik om binêre boodskapseine in digitale videoseine te verskuil. Die doel van die versteekte boodskappe is om die oorspronklike ontvanger van 'n bepaalde digitale video-afskrif te identifiseer. Indien 'n onwettige afskrif van die video ontdek word, kan die oorspronklike ontvanger daarvoor verantwoordelik gehou word. Ons toets die vermoë van die watermerksein om verskeie veranderinge aan die gewatermerkte video te oorleef, byvoorbeeld kompressie, die verwydering van videorame, en amplitudeskalering. Die skema vaar goed in die afwesigheid van videokompressie, maar is nie robuus wanneer die video beduidend saamgepers word nie. Ons implementeer ook 'n dinamiese skema vir verraaier-opsporing, gebaseer op 'n tegniek deur Fiat en Tassa. Die doel van hierdie skema is om groepe gebruikers te identifiseer wat saamwerk om 'n watermerkskema te kul deur gewatermerkte videos te kombineer. Ons toets die verraaier-opsporingskema in 'n simulasie van 'n gewatermerkte video-uitsendingstelsel wat 'n roofuitsending monitor. Wanneer die roofuitsending vertraag is, vind ons dat die voorgestelde verraaier-opsporingskema die skuldige gebruikers vinniger uitwys as die skema van Fiat en Tassa.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations...

- The Naspers Media Lab, for sponsoring this thesis and for providing a fun and interesting environment in which to work on it.
- My supervisor, Prof. Gert-Jan van Rooyen, for all of his help and guidance during this project.
- Dr. Herman Engelbrecht, for his effort making sure that the Media Lab was functional as well as fun.
- My fellow students in the Media Lab, for all the help, advice, and entertainment with which they provided me during my studies.
- Simon Streicher, for helping me to translate the abstract into Afrikaans.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

AWGN	Additive white Gaussian noise
BER	Bit error rate
CDMA	Code division multiple access
CP	Content provider
DCT	Discrete cosine transform
DFT	Discrete Fourier transform
DM	Dither modulation
DRM	Digital rights management
DWR	Document-to-watermark ratio
FPS	Frames per second
FT	Fiat and Tassa
HSV	Hue-saturation-value colour space
OFDM	Orthogonal frequency division multiplexing
QIM	Quantisation index modulation
RGB	Red-green-blue colour space
STDM	Spread transform dither modulation
TV	Television
$Y'C_bC_r$	Luma (Y'), blue-difference (C_b), red-difference (C_r) colour space
Y4M	YUV4MPEG2 uncompressed video format

Variables

b	A watermark message bit
$\hat{b}$	Decoded version of b
b_{delay}	Number of watermark bits transmitted during the delay between the CP broadcast and the pirate rebroadcast
b_{k_i}	The i^{th} watermark message bit encoded in the k^{th} video block
$b_?$	An unreadable watermark message bit

c_k	Number of unique variants transmitted during the k^{th} iteration of the traitor tracing scheme
d_l	Tardos codeword length constant
d_δ	Tardos cut-off constant
d_τ	Tardos score threshold constant
f_h	Fundamental frequency of $h_k[\zeta]$
$\hat{f}_h$	Decoder estimate of f_h
f_s	Sampling frequency of $h_k[\zeta]$
f_Δ	Histogram bin size scaling factor of $h_k[\zeta]$
K	Number of video blocks in the host video signal, $v[x, y, t]$
l	Watermark codeword length
N_b	Number of watermark message bits in a video block
n_{ch}	Scalar channel noise
N_c	Number of pirates carrying out a collusion attack
$\hat{N}_c$	Upper bound on the number of collaborating pirates
N_d	Number of detected pirates
N_l	Number of levels in the dynamic traitor tracing search tree
N_s	Number of coordinates for which $s_{k_i}[x', y', t']$ is non-zero
N_u	Number of users of video content from the CP
r_{conv}	Frame rate to which a watermarked video is converted by a pirate
r_{dec}	Standard video frame rate used for watermark encoding and decoding
r_k	Approximate frame rate scaling factor for the k^{th} video block
$\hat{r}_k$	Decoder estimate of r_k
$\hat{r}'_k$	Decoder tested value of r_k
r_{wm}	Bit rate of watermark signal
s_{k_i}	The i^{th} STDM chip signal in the k^{th} video block
t	Temporal coordinate of video signal, in frames
t'	Temporal coordinate of STDM chip signal
T	Number of frames in the host video signal, $v[x, y, t]$
T_b	Number of video frames in a video block
T_c	Length, in frames, of a cell in a video signal
t_{delay}	Time delay between CP broadcast and pirate rebroadcast
T_s	Length (along time axis) of STDM chip signal

u	A user of the digital video content
u_i	The i^{th} user of the digital video content
x	Horizontal coordinate of video signal, in pixels
x'	Horizontal coordinate of STDM chip signal
X	Width, in pixels, of the host video signal, $v[x, y, t]$
X_a	First colour component of chroma subsampling ratio
X_b	Second colour component of chroma subsampling ratio
X_c	Width, in pixels, of a cell in a video signal
X_J	Luma component of chroma subsampling ratio
X_l	Width, in pixels, of the luma frame of the host video
X_s	Width of the STDM chip signal
y	Vertical coordinate of video signal, in pixels
y'	Vertical coordinate of STDM chip signal
Y	Height, in pixels, of the host video signal, $v[x, y, t]$
Y_c	Height, in pixels, of a cell in a video signal
Y_l	Height, in pixels, of the luma frame of the host video
Y_s	Height of the STDM chip signal
z	A scalar host signal
z_{k_i}	The scalar signal required to encode the i^{th} bit in the k^{th} video block
α	Pixel value scaling factor applied by the pirate
α_k	Pixel value scaling factor applied to the k^{th} video block
$\hat{\alpha}_k$	Decoder estimate of α_k
β_k	Likelihood score of gain estimation
δ	Tardos cut-off parameter
$\delta_{\min}$	Minimum Tardos cut-off parameter
Δ	Quantiser step size
Δ_{bin}	Histogram bin size for $h_k[\zeta]$
ϵ_1	Soundness of a traitor tracing scheme
ϵ_2	Completeness of a traitor tracing scheme
θ_k	Approximate frame offset for the k^{th} video block
$\hat{\theta}_k$	Decoder estimate of θ_k
$\hat{\theta}'_k$	Decoder tested value of θ_k
λ_0	Pixel value that encodes a '0'
λ_1	Pixel value that encodes a '1'
ρ_j	Probability of the j^{th} bit of a Tardos codeword being '1'

ρ'_j	Untransformed probability value for the j^{th} bit of a Tardos codeword
σ_v^2	Variance of the host video signal, $v[x, y, t]$
$\sigma^2_{w^{(u)}}$	Variance of the watermark signal, $w^{(u)}[x, y, t]$
τ	Tardos incrimination threshold
ϕ_1	Dynamic Tardos codeword length optimisation parameter
ϕ_2	Dynamic Tardos codeword length optimisation parameter
ϕ_3	Dynamic Tardos codeword length optimisation parameter

Vectors and Matrices

$\mathbf{m}$	Digital watermark message signal
$\hat{\mathbf{m}}$	Decoded version of $\mathbf{m}$
$\mathbf{m}^{(C)}$	Watermark message present in $p^{(C)}[x, y, t]$
$\mathbf{m}_j^{(C)}$	The j^{th} bit of $\mathbf{m}^{(C)}$
$\mathbf{m}^{(C,k)}$	The collusion-attacked watermark message from the k^{th} iteration
$\mathbf{m}^{(C,k,i)}$	The collusion-attacked watermark message from the i^{th} sub-segment of the k^{th} iteration
$\mathbf{m}^{(u)}$	Watermark message assigned to the user, u
$\mathbf{m}_j^{(u)}$	The j^{th} bit of $\mathbf{m}^{(u)}$
$\hat{\mathbf{m}}^{(u)}$	Decoded version of $\mathbf{m}^{(u)}$
$\hat{\mathbf{m}}_j^{(u)}$	The j^{th} bit of $\hat{\mathbf{m}}_j^{(u)}$
$\mathbf{m}^{(W,k)}$	The watermark message assigned to user set W during the k^{th} iteration
$\mathbf{m}_j^{(W,k)}$	The j^{th} bit of $\mathbf{m}^{(W,k)}$
$\mathbf{m}^{(W,k,i)}$	The i^{th} sub-segment of $\mathbf{m}^{(W,k)}$
$\mathbf{p}(c)$	Dynamic Tardos bit-mask vector for c collaborating pirates
$\mathbf{p}(c)_j$	The j^{th} bit of $\mathbf{p}(c)$
$\mathbf{S}$	Original Tardos score matrix
$\mathbf{S}'$	Symmetric Tardos score matrix
$\mathbf{s}'(W)$	Symmetric Tardos score vector for the user set, W

Functions

$d_{\text{DM}}(b, \Delta)$	Dither value for bit, b, and step size, Δ

$d_l(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2)$	Optimal dynamic Tardos length constant for $\hat{N}_c$ pirates, $\hat{N}_u$ codewords, ϵ_1-soundness, and ϵ_2-completeness
$d_\delta(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2)$	Optimal dynamic Tardos cut-off constant for $\hat{N}_c$ pirates, $\hat{N}_u$ codewords, ϵ_1-soundness, and ϵ_2-completeness
$d_\tau(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2)$	Optimal dynamic Tardos threshold constant for $\hat{N}_c$ pirates, $\hat{N}_u$ codewords, ϵ_1-soundness, and ϵ_2-completeness
$f_b(v, k)$	Partitioning function that gives the k^{th} block of $v[x, y, t]$
$f_{\text{incr}}(\mathbf{m}^{(C)}, M)$	Function that incriminates a set of users based on $\mathbf{m}^{(C)}$
$f_p(v, s)$	STDM projection function
$\tilde{f}_p(v, s)$	STDM projection function of a high resolution host signal onto a low resolution chip signal
$f_r(v[x, y, t])$	Frame de-synchronisation function applied by the pirate
$h_k[\zeta]$	Histogram of the DM values for the k^{th} video block
$H_k[f]$	DFT of $h_k[\zeta]$
$l(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2)$	Optimal dynamic Tardos codeword length for $\hat{N}_c$ pirates, $\hat{N}_u$ codewords, ϵ_1-soundness, and ϵ_2-completeness
$l_{\min}(c)$	Minimum number of bits of $\mathbf{m}^{(W,k)}$ required for $N_c = c$
$n[x, y, t]$	Noise signal
$n_k[x, y, t]$	Noise signal added to the k^{th} video block
p	Received version of z
$p^{(C)}[x, y, t]$	Pirate copy produced by collusion attack
$p^{(u)}[x, y, t]$	Pirate version of $v^{(u)}[x, y, t]$
$p^{(u)}_{k_{\text{sync}}}$	Correctly synchronised pirate version of the k^{th} video block
$\hat{p}^{(u)}_{k_{\text{sync}}}$	Decoder synchronised pirate version of the k^{th} video block
$q(z, \Delta)$	Scalar quantiser with step size, Δ
$q_{DM}(z, b, \Delta)$	DM-encoded value of the host, z, for the bit, b, and step size, Δ
$q_{STDM}(v, b, \Delta, s)$	STDM-encoded value of the host, v for the bit, b, step size, Δ, and chip signal, s
$\tilde{q}_{STDM}(z, b, \Delta, s)$	STDM-encoded value of the high resolution host, v, for the bit, b, step size, Δ, and low resolution chip signal, s
$s[x, y, t]$	Pseudonoise chip signal
$s_{\text{dyn}}(u_i, c, \mathbf{m}^{(C)})$	Dynamic Tardos incrimination score for user u_i, for c collaborating pirates, based on $\mathbf{m}^{(C)}$
$s_{\text{incr}}(u_i, \mathbf{m}^{(C)})$	Tardos incrimination score for user u_i, based on $\mathbf{m}^{(C)}$

$s'_{\text{incr}}(u_i, \mathbf{m}^{(C)})$	Symmetric incrimination score for user u_i, based on $\mathbf{m}^{(C)}$
$s_{k_i}[x, y, t]$	STDM chip signal for the i^{th} bit of the k^{th} video block
$u(t)$	Unit step function
$v[x, y, t]$	Host video signal
$v_k[x, y, t]$	The k^{th} block of the host video signal
$\tilde{v}_k[x, y, t]$	The downsampled version of the k^{th} block of the host video signal
$v^{(u)}[x, y, t]$	Watermarked version of $v[x, y, t]$, assigned to the user, u
$v_k^{(u)}[x, y, t]$	The k^{th} block of the watermarked version of $v[x, y, t]$, assigned to the user, u
$w^{(u)}[x, y, t]$	Watermark signal that encodes the message, $\mathbf{m}^{(u)}$
$w_{k_i}^{(u)}[x, y, t]$	Watermark signal that encodes the i^{th} bit in the k^{th} video block
$\tilde{w}_{k_i}^{(u)}[x, y, t]$	Downsampled version of the watermark signal that encodes the i^{th} bit in the k^{th} video block
$w_k^{(u)}[x, y, t]$	The k^{th} block of the watermark signal, $w_{k_i}^{(u)}[x, y, t]$
$z_{k_i}(\hat{r}'_k, \hat{\theta}'_k)$	The scalar DM value calculated for $\hat{r}'_k$ and $\hat{\theta}'_k$
$\alpha(t)$	Pixel value gain function applied by the pirate
$\hat{\alpha}_k(\hat{r}'_k, \hat{\theta}'_k)$	Decoder estimated gain for $\hat{r}'_k$, and $\hat{\theta}'_k$
$\beta_k(\hat{r}'_k, \hat{\theta}'_k)$	Likelihood score for $\hat{r}'_k$, and $\hat{\theta}'_k$
$\delta[x', y', t']$	Kronecker delta
$\delta(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2)$	Optimal dynamic Tardos cut-off parameter for $\hat{N}_c$ pirates, $\hat{N}_u$ codewords, ϵ_1-soundness, and ϵ_2-completeness
$\tau(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2)$	Optimal dynamic Tardos score threshold for $\hat{N}_c$ pirates, $\hat{N}_u$ codewords, ϵ_1-soundness, and ϵ_2-completeness

Sets

C	The set of pirates carrying out a collusion attack
$\hat{C}$	The set of users incriminated as pirates
D	The set of users disconnected by the CP
G_p	The set of user sets known to contain at least one pirate
I	The set of users assumed to be innocent
L_k	The "left" subset of users in the k^{th} branch of the Fiat and Tassa search tree
P	The set of users to whom a segment received from a pirate broadcast was originally transmitted

P_{FT}	The set of disjoint user sets into which U is partitioned in the Fiat and Tassa scheme
P_k	The set of user sets incriminated by the watermark message, $\mathbf{m}^{(W,k)}$
R	The set of possible frame rate scaling factors tested by the decoder
R_k	The "right" subset of users in the k^{th} branch of the Fiat and Tassa search tree
S_p	The set of user sets suspected of containing at least one pirate
U	The set of all users of video content from the CP
V_C	The set of watermarked videos available to the pirates
W_k	The set of user sets suspected of containing at least one pirate at the beginning of iteration k
W'_k	The set of users that does not contain a detected a pirate at the beginning of iteration k
Θ	The set of possible video block offsets tested by the decoder
Ω_k	The set of all possible coordinates of an STDM chip signal
Ω_{k_i}	The set of coordinates for which $s_{k_i}[x', y', t']$ is non-zero
Ω'_{k_i}	The set of coordinates, chosen by the encoder, for which $\tilde{w}^{(u)}_{k_i}[x', y', t']$ is non-zero

Transforms

$D\{\cdot\}$	Downsample a video block to the resolution of the ship signal

Operators

$\lvert\cdot\rvert$	Absolute value or size of a set
$\lfloor\cdot\rfloor$	Integer floor
$\arg\max(\cdot)$	Argument that yields the maximum value
$\arg\min(\cdot)$	Argument that yields the minimum value
$\ln(\cdot)$	Natural logarithm
$\max(\cdot)$	Maximum value
$O(\cdot)$	Big-O notation
$P(\cdot)$	Probability of
$\text{round}(\cdot)$	Nearest integer
sgn	Signum function

weight$(\cdot)$	Number of non-zero function values

Chapter 1

Introduction

In a digital video distribution system, a content provider (CP) legally distributes proprietary digital video content to a number of authorised users who pay to access the content. Examples of such systems include pay-TV, online content stores, and DVD distribution. Digital rights management (DRM) schemes allow a CP to grant content access to authorised users, and deny access to anyone else. An authorised user might access content, make a copy of it, and redistribute the copy to non-paying, non-authorised users. This illegal redistribution of proprietary content is commonly known as piracy, and the user responsible for the redistribution is referred to as a pirate. Non-paying users who have access to a pirate copy may be reluctant to pay the CP for content that is available (illegally) for free. It is therefore in the interest of the CP to prevent piracy of its content.

1.1 Motivation

While conventional DRM schemes can be effective at preventing unauthorised users from accessing content, they have a limited ability to prevent unauthorised redistribution of content once it has been accessed [1]. If an authorised user is able to view some video content, then that user can find ways to record that content while it plays and redistribute the recording to unauthorised users [2]. A solution to this form of piracy is to make such a recording traceable to the pirate, so that, upon finding a pirate copy of its content, the CP can take action against the pirate to prevent further piracy.

In this thesis, a digital watermarking method is presented that can be used to embed hidden information into digital video content that is intended to remain present in any usable copy of that content. A CP can use this method to embed unique, user-identifying information into the video content it provides to each authorised user. A pirate copy can be traced to the pirate by retrieving the hidden information present in the copy. A determined pirate may attempt to destroy the incriminating information by modifying the pirate

copy before redistributing it. Therefore, the watermarking scheme is designed so that the hidden information can be retrieved after modifications are made to the video signal. The robustness of the scheme is tested to determine how difficult it is to remove the hidden information.

1.2 Background

Digital watermarking is the process of modifying a signal (the host) in such a way that, to a human user, the modified signal is indistinguishable from the original, while, using a certain technique, these modifications can be decoded to reveal some hidden message signal. In this thesis we focus on watermarking digital video signals with binary messages. We embed digital watermarks in *uncompressed* digital video with the intention that the watermark signals remain present after the watermarked videos are compressed using any compression algorithm. Watermarking schemes exist that are specific to particular compressed digital video formats. However, their usefulness is limited to the case where videos remain in their original compressed formats [3, 4].

1.2.1 Communication Model of a Watermarking Scheme

Figure 1.1 shows a digital watermarking scheme, modelled as a communication system for the hidden message signal. We represent the binary watermark message signal as a row vector, $\mathbf{m}^{(u)}$, of bits, such that the j^{th} bit of $\mathbf{m}^{(u)}$ is $\mathbf{m}_j^{(u)}$. The message, $\mathbf{m}^{(u)}$, uniquely identifies the user, u, who receives the watermarked video. The watermarking scheme must ensure that if u distributes a pirate copy of a video, the decoder can reproduce $\mathbf{m}^{(u)}$ (or a good enough approximation of $\mathbf{m}^{(u)}$) from that copy so that u can be incriminated with certainty.

The encoder takes $\mathbf{m}^{(u)}$ and the host signal, v, as input, and outputs a watermark signal, $w^{(u)}$. This watermark is added to v to give the watermarked signal, $v^{(u)}$, which is distributed to a particular user, u. The channel noise, n, represents the distortions caused by any processes to which $v^{(u)}$ is subjected by the user before it is redistributed as the pirate copy, $p^{(u)}$. When $p^{(u)}$ is found, it is input to the decoder, which decodes from it the message, $\hat{\mathbf{m}}^{(u)}$. Since $p^{(u)}$ is a pirate copy of $v^{(u)}$, the watermarking scheme is successful if there is enough information in $\hat{\mathbf{m}}^{(u)}$ to identify u. In this thesis we will not cover the details of the method by which the CP distributes $v^{(u)}$ to the user or the method by which the CP obtains $p^{(u)}$.

In the digital watermarking scheme depicted in Figure 1.1, the unmarked host signal, v, is not available to the decoder. This is known as a *blind* or *oblivious* watermarking scheme. Having v available at the decoder simplifies watermark detection, as well as the reversal of the processes applied in producing $p^{(u)}$ from $v^{(u)}$ [5, 6]. However, the requirement that v be available at the

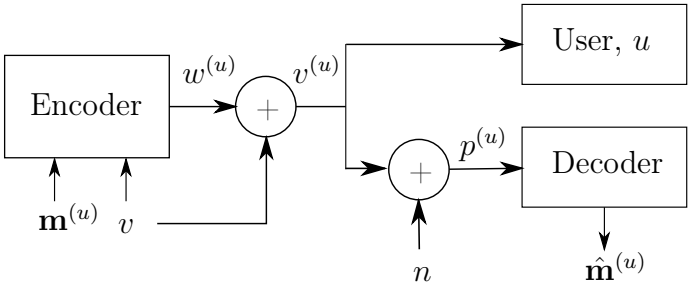


Figure 1.1: Model of a blind digital watermarking scheme as a communication system. The watermark message, $\mathbf{m}^{(u)}$, is encoded into the watermark signal, $w^{(u)}$, which is added to the host signal, v, to give the watermarked signal, $v^{(u)}$. The user, u, receives $v^{(u)}$, modifies it by adding a noise signal, n, and redistributes the pirate copy, $p^{(u)}$. Upon finding $p^{(u)}$, the watermark decoder is used to determine an estimate, $\hat{\mathbf{m}}^{(u)}$, of the watermark message embedded in $p^{(u)}$.

decoder limits watermark decoding to devices where a database of unmarked content can be stored securely [7]. Blind decoding, on the other hand, can be implemented on devices in the possession of users without the risk of exposing unmarked content [8]. Due to this more widespread usefulness, in this thesis we consider only blind watermarking schemes.

1.2.2 Properties of Digital Watermarks

An important property of digital watermarks is *security*. We refer to a digital watermarking scheme as being secure if a pirate cannot encode or decode a watermark, or modify the message $\mathbf{m}^{(u)}$ at will, without knowledge of some secret known to authorised encoders and decoders. This type of security is similar to that of a digital signature scheme, where the receiver of a message signed with a valid digital signature can be certain that the message was generated by a known sender, and was not modified since it was transmitted by that sender [9]. A secure watermarking scheme ensures that the CP does not take action against innocent users, based on watermarks that have been forged by the actual pirates.

A digitally watermarked signal should be perceptually similar to the unmarked signal. We refer to this property as the *fidelity* of the watermarking scheme. Noticeable distortion in the content caused by low-fidelity watermarking may reduce users' satisfaction with, and, possibly, their willingness to pay for, content distributed by the CP [10]. It is difficult to objectively measure the perceptibility of the distortion caused by digital watermarking, since perceptibility depends on the physiology of the viewer. However, we can measure the signal distortion caused by the watermark by comparing the relative signal power of the host to that of the watermark signal. This metric is known as the document-to-watermark ratio (DWR) [11, 6, 12], and is given by

$$\text{DWR} = \frac{\sigma_v^2}{\sigma_{w^{(u)}}^2}, \tag{1.2.1}$$

where σ_v^2 is the variance of the host signal, v, and $\sigma_{w^{(u)}}^2$ is the variance of the watermark signal, $w^{(u)}$. A high DWR indicates low levels of distortion of the host signal, which should result in reduced quality degradation. However, there exists a trade-off between fidelity and another important property of digital watermarks: *robustness*.

We use the term "robustness" to describe the ability of a watermark signal to resist decoding errors as the distortion applied to $v^{(u)}$ in producing $p^{(u)}$ increases. If the identifying information in a watermarked video cannot be decoded reliably due to distortion, then no action can be taken against the pirate. Ideally, a watermark should be robust enough to be decoded correctly from a pirate copy, as long as the quality of that copy is still acceptable for use. That is, the watermark need not be robust against distortions that degrade the pirate copy to the point that it is of no commercial value [13].

Correct decoding of the watermark message, $\mathbf{m}^{(u)}$, relies on two factors. Firstly, the bits of $\mathbf{m}^{(u)}$ embedded in $v^{(u)}$ must remain uncorrupted in $p^{(u)}$. Secondly, the decoder must decode the value of each message bit from the location within the video signal in which that bit was embedded by the encoder. We refer to this second condition as *synchronisation*. For video watermarks, both temporal and spatial synchronisation is necessary. This can be achieved by embedding, along with the watermark message, a known signal for which the decoder searches. This signal, known as a *template*, is then used by the decoder to reverse any temporal and spatial manipulations that were applied to $w^{(u)}$ [14, 15]. A disadvantage of using templates is that the template itself can be destroyed, preventing the decoder from synchronising [7]. When considering the robustness of a watermarking scheme it is necessary to take into account the nature of the distortion applied to $v^{(u)}$. A digital watermarking scheme may be highly robust against some forms of distortion, yet weak against others [10].

1.2.3 Attacks on Digital Video Watermarks

Typical video processing operations that may be carried out on watermarked videos include lossy compression, resizing, and frame rate conversion. These operations may be carried out by the CP itself prior to distributing watermarked videos, or by pirates before redistributing pirate copies. In addition to these operations, pirates may subject watermarked videos to processes intended to corrupt the watermark messages, or to desynchronise the encoder and decoder, so that correct decoding and incrimination cannot occur. These processes may include addition of noise to the video, scaling of pixel values, deletion of frames, and frame cropping. In this thesis we refer to both of the above classes of video manipulations as attacks.

While a single pirate can carry out any of the attacks mentioned above, a group of collaborating pirates can carry out a *collusion attack*, where the group combines its members' differently watermarked copies of the same content into a new pirate copy. This attack is intended to produce a copy that bears either a corrupted watermark, which does not identify any of the pirates, or a valid watermark that incriminates an innocent user. If the latter is feasible for a particular watermarking scheme, then a pirate incriminated by the presence of a watermark in a pirate copy could reasonably claim to have been falsely incriminated in this way.

1.2.4 Traitor Tracing Schemes

Traitor tracing schemes construct watermark messages in such a way that a pirate copy produced by collusion can be traced with some certainty to at least one pirate collaborator, while false incrimination is unlikely. We refer to these message constructions as collusion-secure codes. In a *static* traitor tracing scheme, collusion-secure codes are constructed prior to content distribution, using an estimated upper bound on the number of collaborating pirates. When a pirate copy is found, users are incriminated based on the composition of the modified code present in the copy. Static schemes provide no certainty that *all* pirates will be incriminated, and the ability of static schemes to incriminate pirates with certainty is compromised if the number of collaborating pirates is underestimated [16, 17].

If real-time feedback from a pirate redistribution is available, as is the case with illegal live rebroadcast of internet multicast or pay-TV content, a *dynamic* traitor tracing scheme may be used [18, 19, 16]. In a dynamic scheme, segments of content are digitally watermarked on the fly, based on previous watermarked segments received from a pirate redistribution. Dynamic schemes are capable of updating estimates of the number of collaborators and immediately disconnecting incriminated pirates until the pirate redistribution ceases [19].

1.3 Related Work

Several digital watermarking techniques have been developed for digital video, some of which are compared in [20] and [7]. Many of these techniques transform the host signal to another domain, embed the watermark in the transformed host, and then apply the inverse transform to produce the watermarked video. Transform domains used in this way include the discrete Fourier transform (DFT) [15], discrete cosine transform (DCT) [21, 22, 23], and wavelet [24, 25] domains. Domain transformation can make the watermark inherently robust against certain attacks [15], but the transformation and inverse transformation steps can be computationally intensive. Other techniques add the watermark

signal directly to the host signal [14]. We refer to this type of technique as *spatial domain* watermarking.

Several works have focussed on collusion attacks. Chor et al. introduced traitor tracing schemes in [26] as a means of tracing pay-TV users who combine decryption keys from their decoders to produce pirate decoders. In [27], Boneh and Shaw introduced collusion-secure codes for digital watermarks. Fiat and Tassa introduced dynamic traitor tracing schemes in [18], and in [28] Tassa combined a scheme from [18] with the code construction of Boneh and Shaw to trace collaborating pirates using digitally watermarked broadcast content. Collusion-secure code constructions based on channel multiple-access techniques are presented in [29, 30], and some other constructions are given in [31, 32]. An alternative approach to protecting digitally watermarked images is used in both [33] and [34]. In these schemes, each watermarked copy is slightly warped in a unique way so that combining differently watermarked copies results in a degraded pirate copy.

1.4 Objectives

The first objective of this thesis is to develop a prototype digital watermarking scheme for uncompressed digital video. The watermarking scheme should consist of an encoder that can embed arbitrary binary watermark messages into a host video signal, and a decoder that is capable of reproducing the watermark message embedded in a given watermarked video signal. The robustness of the digital watermarking scheme will be tested in terms of bit error rate (BER) of decoded watermark messages when the watermarked video is subjected to the following attacks:

1. addition of white Gaussian noise,
2. scaling of pixel values by a constant gain factor,
3. random frame deletion,
4. frame rate modification,
5. spatial resizing,
6. frame cropping, and
7. lossy compression.

The second objective of this thesis is to implement a dynamic traitor tracing scheme that is capable of tracing all members of a group of pirates who carry out a collusion attack on a watermarked video.

1.5 Contributions

We propose and evaluate a method of achieving temporal synchronisation in a digital video watermarking scheme without the use of a template signal. We show that this method is capable of achieving synchronisation when the frame rate of the watermarked video is changed, and when individual frames are randomly deleted. We also propose a modification to the dynamic traitor tracing scheme of Fiat and Tassa that makes the scheme applicable to dynamic settings with delayed feedback from pirate rebroadcasts. We show that with this modification the scheme traces all traitors in less time than the original scheme.

1.6 Overview

In this thesis we describe the design and implementation of a prototype digital video watermarking scheme and dynamic traitor tracing scheme. Our designs are based on existing designs of watermarking and traitor tracing schemes, which we review in Chapter 2. In Chapter 3 we describe the designs of our schemes, and in Chapter 4 we provide details of the implementation of the designs in software. We test the watermarking scheme by watermarking several uncompressed digital videos, subjecting the watermarked videos to the attacks listed in Section 1.4, and determining the BER of the messages decoded from the attacked videos. We test the traitor tracing scheme by simulating dynamic settings with pirate rebroadcasts. Details of the tests conducted and their results can be found in Chapter 5. In Chapter 6 we conclude and discuss possibilities for future work on this topic.

Chapter 2

Literature Review

In this chapter we review the literature on which our designs are based. We can divide the work into two basic components. The first component is the digital watermarking scheme that embeds the bits of a given binary message signal into a host video signal. The second component is the traitor tracing scheme that generates message signals for each user's watermarked copy in a way that makes the entire scheme robust against collusion attacks. Since these two components are often studied in isolation, we review the literature associated with each component in separate sections.

2.1 Digital Watermarking

We begin by discussing literature focussed on digital watermarking. We chose to implement a spatial domain watermarking scheme because of the lower computational requirement, when compared to transform domain watermarking. This makes spatial domain watermarking more suitable for use where watermark encoding may need to be performed on limited hardware such as a TV set-top box. Therefore, we will not discuss transform domain techniques, such as those used in [15, 21, 25].

2.1.1 The Uncompressed Digital Video Signal

An uncompressed digital video consists of a sequence of T frames, each of which is a 2-dimensional pixel array of width X and height Y. Each pixel in a colour video can be represented by a coordinate vector in a colour space. Examples of colour spaces are RGB, HSV, and $Y'C_bC_r$ [35]. For our implementation we chose to represent digital videos using the $Y'C_bC_r$ colour space. We motivate this choice in Chapter 4.

The three colour components of $Y'C_bC_r$ are the luma (Y'), blue-difference (C_b), and red-difference (C_r) components. The luma component describes the brightness of the pixel, such that a video or image containing only this com-

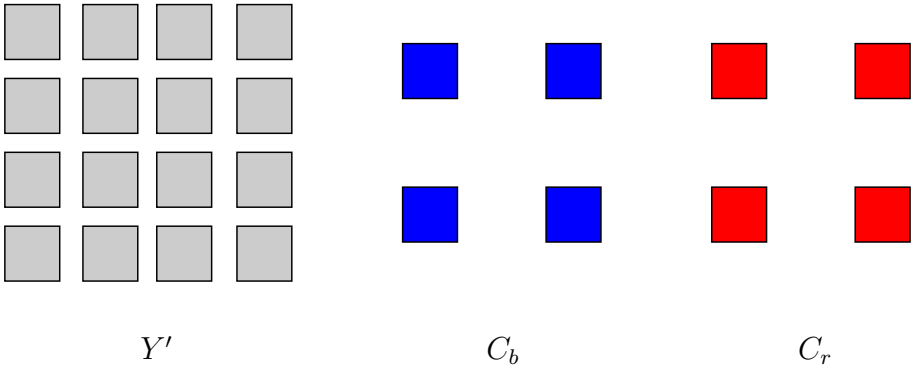


Figure 2.1: A 4×4 pixel video frame represented using the $Y'C_bC_r$ colour space with 4:2:0 chroma subsampling. Each block represents a byte-valued pixel.

ponent would appear to be greyscale or "black-and-white". The blue-difference and red-difference components provide additional colour information for each pixel so that the pixel colour can be reproduced from all three components.

The separation of luma and colour channels provided by the $Y'C_bC_r$ representation allows video and image compression algorithms to make use of a technique known as *chroma subsampling*. Chroma subsampling involves using a lower spatial resolution for colour information (C_b and C_r channels) than for brightness information (luma channel) in an image or video. This technique takes advantage of the fact that the human eye is less sensitive to changes in colour than to changes in brightness, in order to reduce the space required to store an image or video [7]. The type of chroma subsampling used by a digital image or video format is described by a ratio of the form $X_J : X_a : X_b$, where X_J is usually equal to 4. For each colour channel, $X_J : X_a$ represents the ratio of luma pixels to colour pixels for every odd row of the image (video frame), and $X_J : X_b$ represents the ratio of luma pixels to colour pixels for every even row of the image (video frame).

Before an image or video is displayed, the subsampled colour channels are upsampled to the resolution of the brightness channel. The three colour coordinates corresponding to each pixel are then converted to the RGB primary values necessary to recreate the correct pixel colour on the display device. Since the resolution of the displayed image is the same as that of the brightness channel of the stored image, we refer to the resolution of the brightness channel when quoting the resolution of an image or video. Figure 2.1 illustrates the $Y'C_bC_r$ representation of a video frame with a height of 4 pixels and a width of 4 pixels, using 4:2:0 chroma subsampling. Both the height and width of each colour channel are half those of the brightness channel.

For simplicity, Chapters 2 and 3 describe techniques for digitally watermarking a single colour channel of the video signal; i.e. each pixel is considered to be represented by a scalar value. We can then represent a digital video signal as a function, $v[x, y, t]$, where $x \in \{0, 1, \ldots, X-1\}$ and $y \in \{0, 1, \ldots, Y-1\}$ are

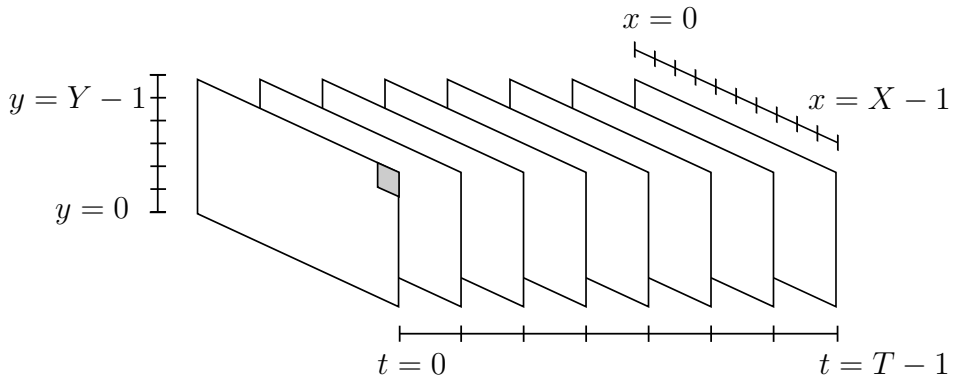


Figure 2.2: Digital video represented as a function, $v[x, y, t]$, where $x \in \{0, \ldots, X-1\}$ and $y \in \{0, \ldots, Y-1\}$ are horizontal and vertical pixel coordinates, respectively, and $t \in \{0, \ldots, T-1\}$ is the frame number. The shaded region represents the single pixel value, $v[X-1, Y-1, 0]$.

horizontal and vertical pixel coordinates, respectively, and $t \in \{0, 1, \ldots, T-1\}$ is the frame number. Since pixel values for a single colour channel are stored as bytes in a digital video, $v[x, y, t] \in \{0, 1, \ldots, 255\}$. Figure 2.2 illustrates this representation of a video signal as a function. In Chapter 4 we describe how we apply the watermarking scheme independently to more than one colour channel.

2.1.2 Spread Spectrum Watermarking

Spread spectrum modulation is a technique used in telecommunications whereby a narrowband signal is transmitted using a significantly wider bandwidth than necessary [36]. The signal power is "spread" over a wide frequency band so that only a fraction of the total signal power occurs within any narrow sub-band. The purpose of this may be to make the transmitted signal difficult to detect in the presence of background noise, or to mitigate the effect of narrowband jamming. Additionally, shared secret keys can be used to modulate and demodulate spread spectrum signals so that unauthorised receivers and transmitters cannot communicate with authorised ones [37, 23]. These properties are analogous to the properties of fidelity, robustness and security that are desirable for digital watermarks [6]. For this reason, several watermarking schemes are based on principles of spread-spectrum communication [37, 13, 21].

The spread-spectrum technique most commonly used for digital watermarking is known as direct sequence spread spectrum (DSSS). Figure 2.3 shows a model of a DSSS communication system. The transmitter generates the DSSS signal by multiplying the modulated message signal by a spreading chip signal. The received DSSS signal is multiplied by the same chip signal prior to demodulation to give the received message. Since it is necessary for the receiver and transmitter to use the same chip signal, the chip signal is a type

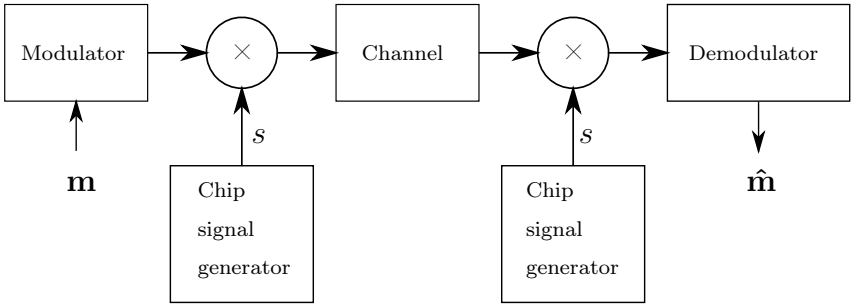


Figure 2.3: Diagram of a direct sequence spread spectrum communication system. The message, $\mathbf{m}$, is first modulated using some narrowband modulation technique. The modulated message signal is then multiplied by a chip signal, s, which spreads the signal over a wide frequency band before transmission. At the receiver the spread spectrum signal is multiplied by s and the resulting narrowband signal is demodulated to give the decoded message signal, $\hat{\mathbf{m}}$.

of shared secret key necessary for communication. The chip signal is chosen to be a periodic, binary polar pseudonoise (PN) sequence with a sharp autocorrelation, which helps the receiver to synchronise on the chip signal [36]. Where multiple PN sequences are used for communication across the same channel, for example in code division multiple access (CDMA) systems [38], the PN sequences are also chosen to have minimal cross-correlation to reduce interference [36].

2.1.3 Host-Aware Watermarking Schemes

In some watermarking schemes, including some of those based on spread-spectrum communication, the message signal, $\mathbf{m}^{(u)}$, is encoded into $w^{(u)}$ independently of the host signal, v [6]. In such a scheme, the watermarked signal $v^{(u)}$ is the sum of a watermark signal, $w^{(u)}$, which can be decoded to give $\hat{\mathbf{m}}$, and the host signal, v, which is of value to a human user, but carries no useful information for the decoder. If blind decoding is used, the decoder cannot simply subtract v from $p^{(u)}$ to find (a possibly distorted version of) $w^{(u)}$. In this setting, the host signal is itself a source of channel noise for the watermark signal, reducing the amount of information that can be communicated reliably within the watermark [39].

Another approach is to encode $\mathbf{m}^{(u)}$ into $w^{(u)}$ in a way that is dependent on v. The scheme in Figure 1.1 on page 3 is of this type, which is sometimes referred to as a *host-aware* scheme. In a host-aware scheme it is possible for the encoder to determine how to distribute the watermarking distortion over $v^{(u)}$ so that the maximum watermarking strength can be used without causing noticeable degradation to the host signal. This can be achieved by using some

type of *perceptual model*, which models how sensitive human senses are to signal distortions in particular parts of the host signal [40]. Once the optimal distribution of distortion is determined, it can be used to scale the strength of $w^{(u)}$ locally in order to maximise the robustness of the watermark, while ensuring high fidelity of the watermarked signal.

2.1.4 Communication with Side Information

In a host-aware scheme, prior to encoding $w^{(u)}$, the encoder knows exactly what "noise" signal (the host signal, v) will be added to $w^{(u)}$ to produce $v^{(u)}$. This prior information about the effect of the transmission channel is known as *side information* [41]. Communication systems where side information is available to the transmitter were first studied by Shannon in [42]. Shannon showed that if the encoder has access to side information, then it can encode $\mathbf{m}^{(u)}$ in such a way that the encoded signal, when added to the known channel noise, yields a signal from which the decoder will correctly produce $\mathbf{m}^{(u)}$.

In [43] Costa proved that, for a noisy channel with a power-constrained transmitter, the capacity of a standard Gaussian channel can be achieved with blind decoding and side information at the transmitter. In order to achieve this capacity, a common codebook must be used for encoding and decoding. The optimal codebook should be constructed so that for each possible state of the side information signal, there exist codewords that are close enough to that state that the transmitter power constraint is obeyed, yet far enough apart to be distinguishable by the decoder.

In [39] Chen and Wornell proposed methods for generating suitable codebooks for digital watermarking where each codeword is a differently quantised version of the host signal. The quantised versions of the host signal are close enough to the original host signal that the power of the watermark signal is constrained, and the decoder can distinguish codewords by finding the nearest quantised version of the received video signal. Their technique is therefore known as *quantisation index modulation* (QIM).

2.1.5 Dither Modulation

One of the QIM techniques presented in [39] generates $v^{(u)}$ by adding to v a dither signal corresponding to $\mathbf{m}^{(u)}$, before quantising v. This technique is referred to as dither modulation (DM). Here we describe the simplest type of DM encoding and decoding, where a single message bit, b, is encoded in the scalar host, z, using a scalar quantiser. Multi-bit messages can be encoded into video signals by partitioning the host video signal into individual scalar values and applying the DM technique to each of these values.

A scalar quantiser, q, maps a value, z, to the nearest point in a set of

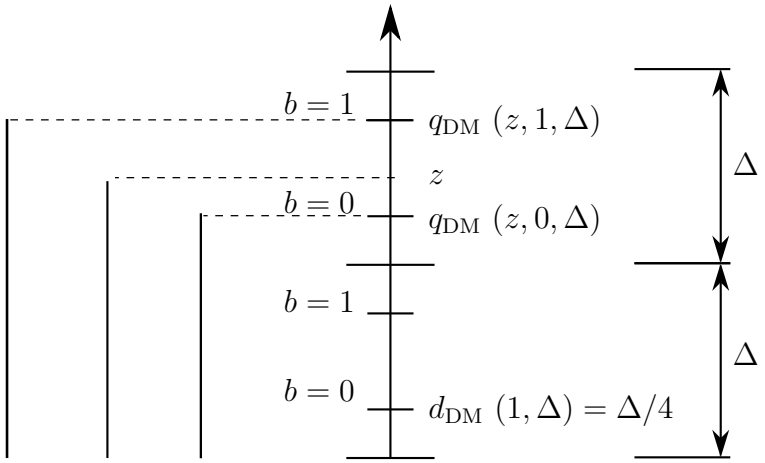


Figure 2.4: Illustration of dither modulation of the scalar host signal z, using a quantiser step size of Δ, and $d_{\mathrm{DM}}(1, \Delta) = \Delta/4$. A binary '1' is encoded into z by generating $q_{\mathrm{DM}}(z, 1, \Delta)$, and a '0' is encoded by generating $q_{\mathrm{DM}}(z, 0, \Delta)$. The maximum possible change induced in the host by DM encoding is $\Delta/2$, which occurs when the value of z corresponds perfectly to a DM encoded bit, b, and the binary compliment of b must be encoded into z.

discrete points, such that

$$q(z, \Delta) = \mathrm{round}(z/\Delta)\Delta, \tag{2.1.1}$$

where Δ is the step size of the quantiser, and round$(\cdot)$ is the function of rounding to the nearest integer. This quantiser can be used in conjunction with a dither signal to embed a bit, $b \in \{0, 1\}$, in the scalar host signal, z, to give

$$q_{\mathrm{DM}}(z, b, \Delta) = q\left[z + d_{\mathrm{DM}}(b, \Delta), \Delta\right] - d_{\mathrm{DM}}(b, \Delta), \tag{2.1.2}$$

where $d_{\mathrm{DM}}(b, \Delta)$ is a dither value corresponding to the bit, b, chosen such that

$$d_{\mathrm{DM}}(1, \Delta) = \begin{cases} d_{\mathrm{DM}}(0, \Delta) + \Delta/2, & d_{\mathrm{DM}}(0, \Delta) < 0 \\ d_{\mathrm{DM}}(0, \Delta) - \Delta/2, & d_{\mathrm{DM}}(0, \Delta) \geq 0 \end{cases}. \tag{2.1.3}$$

In our implementation, we simply choose $d_{\mathrm{DM}}(0, \Delta) = -\Delta/4$ and $d_{\mathrm{DM}}(1, \Delta) = +\Delta/4$. Figure 2.4 illustrates how a scalar host signal, z, can be dither modulated to embed either $b = 0$ or $b = 1$. It can be seen from the figure that the maximum distance between any host signal, z, and the DM value that encodes b is $\Delta/2$, which occurs when z is exactly the DM value that encodes the binary compliment of b. This property results in an upper bound on the amount of watermark signal power required to encode any watermark message into any host signal.

Let n_{ch} be a scalar value representing the channel noise and let $p = q_{\mathrm{DM}}(z, b, \Delta) + n_{\mathrm{ch}}$ be the noisy signal received by the DM decoder. The esti-

mated message bit, $\hat{b}$, decoded from p, is given by

$$\hat{b} = \underset{b' \in \{0,1\}}{\arg\min} \left| y - q_{\mathrm{DM}}(p, b', \Delta) \right|, \tag{2.1.4}$$

where q, Δ, and d_{DM} are the same as those used by the encoder. The decoder will not make an error as long as the magnitude of the channel noise is less than $\Delta/4$. In fact, correct decoding occurs when $2k\Delta - \Delta/4 < n_{\mathrm{ch}} < 2k\Delta + \Delta/4 \, \forall k \in \mathbb{Z}$. However it is impractical to rely on the event where $k \neq 0$.

2.1.6 Spread-Transform Dither Modulation

An extension of DM is introduced in [39] where the host signal is first transformed by projection before encoding and decoding. This technique is referred to as spread-transform dither modulation (STDM). Figure 2.5 illustrates the STDM encoding procedure. Again, we describe a simple case where only a single-bit watermark message is used. Therefore only two watermarked versions of v, $v^{(0)}$, and $v^{(1)}$ are generated, and assigned to users, u_0 and u_1, respectively. The encoding and decoding procedures can simply be repeated on different parts of a video signal to embed multi-bit messages.

The first step of the encoding procedure is to project the host signal, v, onto a pseudo-noise chip signal, s. The projection function, f_p, produces a scaled magnitude of this projection, and is given by

$$f_p(v[x,y,t], s[x,y,t]) = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \sum_{t=0}^{T-1} s[x,y,t] v[x,y,t]. \tag{2.1.5}$$

The nearest value to $f_p(v,s)$ that encodes a bit, b, is determined by applying the scalar DM encoding function in (2.1.2) to $f_p(v,s)$ to give

$$q_{\mathrm{STDM}}(v, b, \Delta, s) = q_{\mathrm{DM}}\left[f_p(v,s), b, \Delta \right]. \tag{2.1.6}$$

The encoder then calculates the watermark signal, $w^{(b)}$, corresponding to b, which is the smallest signal satisfying $f_p(v+w, s) = q_{\mathrm{STDM}}(v, b, \Delta, s)$. This watermark signal is given by

$$w^{(b)}[x,y,t] = \frac{s}{\mathrm{weight}(s)} \left[q_{\mathrm{STDM}}(v, b, \Delta, s) - f_p(v,s) \right], \tag{2.1.7}$$

where

$$\mathrm{weight}(s) = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \sum_{t=0}^{T-1} s[x,y,t]. \tag{2.1.8}$$

The watermark signal is added to the original host signal to give the watermarked signal,

$$v^{(b)}[x,y,t] = v[x,y,t] + w^{(b)}[x,y,t]. \tag{2.1.9}$$

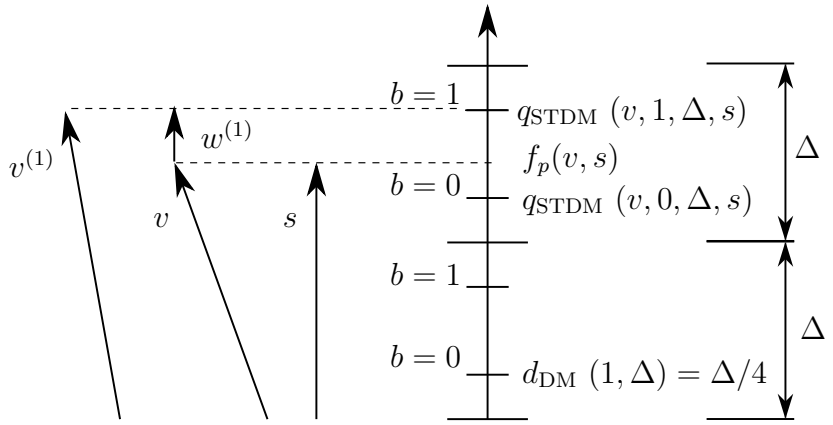


Figure 2.5: Illustration of spread-transform dither modulation of the host signal, v, using a quantiser step size of Δ, and $d_{\text{DM}}(1, \Delta) = \Delta/4$. The host signal, v, chip signal, s, watermark signal, $w^{(1)}$, and watermarked signal, $v^{(1)}$, are represented here as vectors. The encoder first determines the scalar projection of v onto s using the projection function, f_p. The encoder then adds $w^{(1)}$ to v to give $v^{(1)}$, such that the (scalar) projection of $v^{(1)}$ onto s is equal to $q_{\text{STDM}}(v, 1, \Delta, s)$.

Let $p^{(b)} = v^{(b)} + n$ be the noisy watermarked signal received by the decoder. The decoder determines an estimate, $\hat{b}$, of the received message bit as shown in Figure 2.6, by first applying the projection function, f_p, to $p^{(b)}$, using the same chip signal, s, and then applying the DM decoding rule to the resulting magnitude, which gives

$$\hat{b} = \underset{b' \in \{0,1\}}{\arg\min} \left| f_p(p^{(b)}, s) - q_{\text{STDM}}(p^{(b)}, b', \Delta, s) \right|. \tag{2.1.10}$$

The use of a pseudo-noise chip signal has two advantages, which are similar to the advantages provided by spread spectrum techniques. Firstly, it allows large quantisation step sizes to be applied to the projected host signal while only low-amplitude distortions are induced in the watermarked signal [39]. Secondly, when noise is added to the watermarked signal, it is the magnitude projection of that noise onto the chip signal that affects the determination of the bit value at the decoder [39]. Therefore, noise added to the watermarked signal by an attacker has a large effect on decoder errors when the magnitude of its projection onto s is large, but a small effect when that magnitude is small. This property is illustrated in Figure 2.6. If s is kept secret then the attacker wishing to defeat the watermarking scheme by adding noise must either try to guess s (which may be infeasible) or simply add a high level of noise power to try and increase the chance of decoder errors. The latter option is likely to severely degrade the pirate copy.

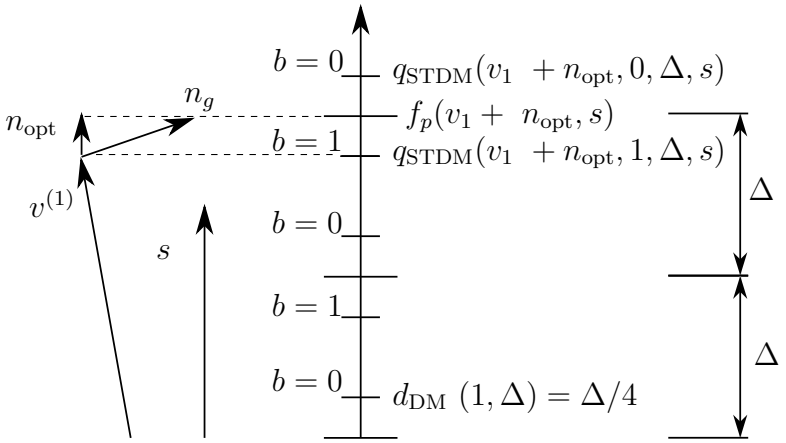


Figure 2.6: Illustration of the effect of an added noise signal on STDM decoding. The chip signal, s, watermarked signal, $v^{(1)}$, and noise signals, n_{opt} and n_g are represented as vectors. The watermarked signal encodes the bit, $b = 1$. This bit is correctly decoded when $f_p(v^{(1)}, s)$ is closer to $q_{\text{STDM}}(v, 1, \Delta, s)$ than to $q_{\text{STDM}}(v, 0, \Delta, s)$. The smallest noise signal required to induce a decoding error is n_{opt}. Since determining n_{opt} requires knowledge of the secret chip signal, s, the pirate adds a random noise signal, n_g. This signal needs to be much larger than n_{opt} in order to induce decoder errors.

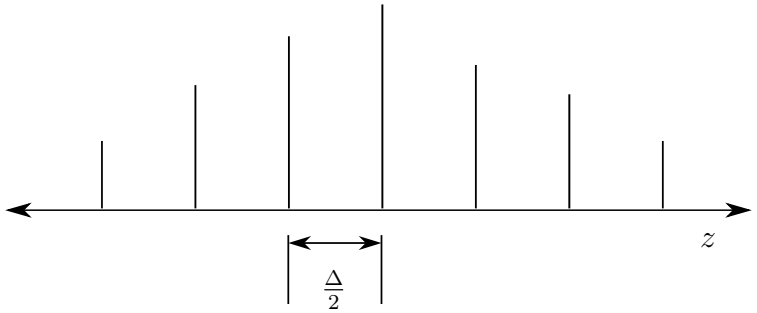


Figure 2.7: Sample histogram of the dither modulated values in $v^{(u)}$, using a quantiser step size of Δ. For each possible scalar value, z, $q(z, \Delta)$ is an integer multiple of Δ, and since $q_{\text{DM}}(z, 0, \Delta)$ and $q_{\text{DM}}(z, 1, \Delta)$ differ by $\Delta/2$, the possible dither modulated values in $v^{(u)}$ are separated by exactly $\Delta/2$.

2.1.7 Gain Attacks

A watermarked signal may be subjected to amplitude scaling. In video watermarking, this can result from brightness or colour adjustments, or from intentional attacks by pirates on embedded watermarks. We refer to amplitude scaling as a *gain attack*. Some watermarking schemes, especially those based on additive spread spectrum and transform domain techniques, are inherently robust against this form of attack [15, 21]. QIM watermarking schemes, on the other hand, are particularly vulnerable to gain attacks [12, 44].

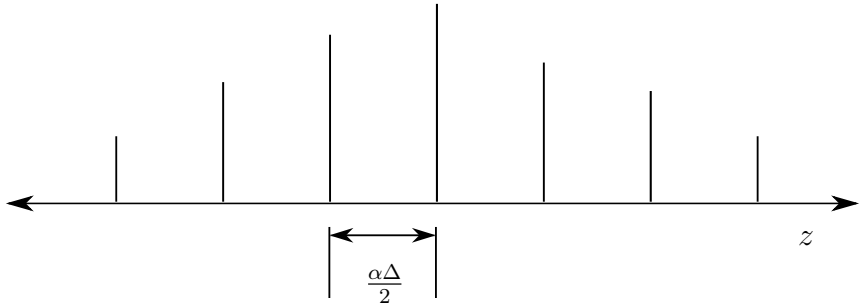


Figure 2.8: Sample histogram of the dither modulated values in $\alpha v^{(u)}$. Since $v^{(u)}$ has been perfectly scaled by the gain factor, α, the dither modulated scalar values it contains are now separated by $\alpha\Delta/2$.

We can see why this is so by considering a DM scheme where a long binary message, $\mathbf{m}^{(u)}$, is encoded into the host signal, v, using a quantiser with step size Δ, to generate the watermarked signal, $v^{(u)}$. If we construct a histogram of the dither modulated scalar values in $v^{(u)}$ it would look similar to the one in Figure 2.7. Let $v^{(u)}$ be subjected to a gain attack that multiplies $v^{(u)}$ by a constant gain, α, producing the signal, $\alpha v^{(u)}$. This new signal is equivalent to the dither modulated signal that would be generated if $\mathbf{m}^{(u)}$ were embedded in the host signal, αv, using a quantiser step size of $\alpha\Delta$. The histogram of the dither modulated values in $\alpha v^{(u)}$ would be similar to those in Figure 2.8. The encoder step size, Δ, is known to the decoder and used in the decoding algorithm. The mismatch between the quantisation step size of $\alpha v^{(u)}$, $\alpha\Delta$, and the step size used by the decoder, Δ, results in decoding errors [12].

In [12] Pérez et al. proposed a technique known as rational dither modulation where the DM encoding procedure is applied to a rational function of the host signal, which is invariant to constant amplitude scaling. The decoder applies the same rational function to the received watermarked signal before applying the DM decoding algorithm. A drawback to this method is that the rational function is not invariant under additive noise. This can cause the decoder-calculated value of the rational function to differ from the encoder-calculated value, resulting in decoding errors.

Another approach to improving the robustness of DM against gain attacks is to let the decoder estimate the gain applied to the watermarked signal and scale the quantisation step size used for decoding accordingly. By observing a histogram of the dither modulated values in $p^{(u)}$, the decoder can estimate $\alpha\Delta$. This is trivial in the case where $p^{(u)} = \alpha\Delta$, and the histogram is similar to the one in Figure 2.8. However, estimating $\alpha\Delta$ can be more difficult when $p^{(u)} = \alpha v^{(u)} + n$, where n is an added noise signal that causes deviation from the perfectly dither modulated values. If n is Gaussian then the noisy dither modulated values become Gaussian distributions around their corresponding perfectly dither modulated means and the histogram may look more like the one in Figure 2.9 [44]. Noting that this distribution appears roughly sinu-

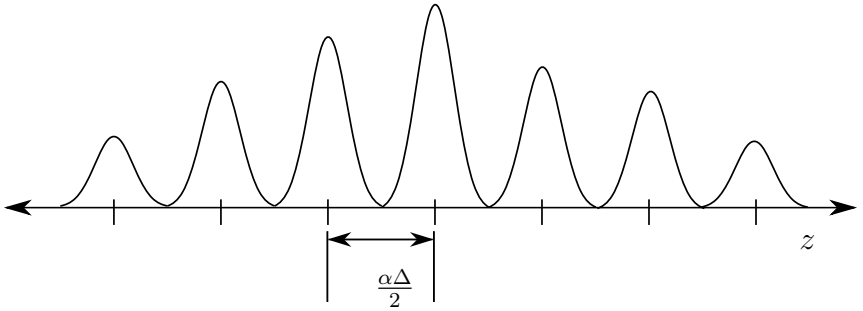


Figure 2.9: Sample histogram of the dither modulated values in $p^{(u)} = \alpha v^{(u)} + n$, where n is a Gaussian noise signal. The scalar values in $p^{(u)}$, from which the decoder must decode watermark message bits, are now distributed around the perfectly encoded values in $\alpha v^{(u)}$. This makes estimation of α more difficult than when no noise is added to $\alpha v^{(u)}$. If the noise is small enough compared to Δ, the histogram should still contain peaks at intervals of $\alpha\Delta/2$. Since the peaks and troughs of the histogram give it a roughly sinusoidal form, the decoder can use the discrete Fourier transform to determine the period of the peaks, and determine α.

soidal, Eggers et al. presented a technique in [44] whereby the fundamental period of the histogram of the dither modulated values of $p^{(u)}$ is determined using the discrete Fourier transform. The fundamental period is then used to calculate $\alpha\Delta$. In [44] the encoder adds a template signal to the host, which is used specifically for gain estimation. In [45] a similar Fourier transform-based approach is used, but without the need for a template signal.

2.2 Traitor Tracing

We now turn our attention to collusion attacks, and methods that have been developed to make digital watermarking schemes robust against these attacks. In this thesis we only consider a single set of collaborating pirates, each of whom have access to content distributed by the CP. The traitor tracing schemes described here can simply be repeated for each disjoint set of collaborating pirates responsible for a particular pirate redistribution.

2.2.1 Collusion Attacks

Let U be the set of users of the CP, where $|U| = N_u$, and let $\mathbf{m}^{(u_i)}$ be the watermark message assigned to user $u_i \in U$, where $i \in \{1, \ldots, N_u\}$. Let $C \subset U$ be the set of N_c collaborating pirates, and let V_C be the set of N_c watermarked video copies to which the pirates in C have access.

2.2.1.1 Averaging Attack

Let $p^{(C)}$ be the pirate copy produced by a collusion attack carried out by C. A simple way in which the pirates in C may attempt to corrupt the watermark signals embedded in their copies is to generate $p^{(C)}$ by "averaging out" their copies so that

$$p^{(C)}[x, y, t] = \frac{\sum_{v \in V_C} v[x, y, t]}{N_c}. \tag{2.2.1}$$

This process is intended to cause interference between the watermark signals, which are unique to each copy, and may, for some watermarking schemes, result in a copy that is more similar to the original host signal than either of the watermarked copies. An advantage of this attack is that it can be carried out with no knowledge of the watermarking scheme.

The problem of successfully communicating watermark messages under this form of collusion attack is analogous to the problem of channel multi-access in telecommunication systems. Techniques like code division multiple access (CDMA) and orthogonal frequency division multiplexing (OFDM) are used in telecommunications systems to allow multiple users to communicate on the same channel at the same time while the interference between their signals is minimal [36, 38]. For this reason these techniques have been applied to some watermarking schemes in order to provide robustness against the averaging attack [29].

2.2.1.2 Bit Selection Attack

Another collusion strategy is to identically partition each video in V_C into parts, and choose a part from one of the videos to be the corresponding part of $p^{(C)}$. The parts may simply be individual frames. However, if the watermarking scheme embeds individual bits across multiple frames, this may not be sufficient to prevent the decoder from correctly decoding a watermark message incriminating a member of C. The pirates may therefore attempt to identify the parts of each video in which individual bits of $\mathbf{m}^{(u)}$ are encoded, and construct $p^{(C)}$ from these parts. The pirates in C may use any strategy for selecting which video contributes each part (encoded message bit) to $p^{(C)}$. Possible strategies include uniform random selection from V_C, or majority selection, where the most commonly occurring version of each part is selected.

Let us assume that the underlying watermark modulation scheme embeds the bits of $\mathbf{m}^{(u)}$ in such a way that there is negligible probability that a pirate in possession of a single watermarked copy can deduce the part of the watermarked copy in which each bit of the watermark message is encoded. Let us assume further that a group of collaborating pirates in possession of multiple watermarked copies can only identify a part corresponding to a particular bit of $\mathbf{m}^{(u)}$ by finding a part that occurs as an encoded '1', and as an encoded '0' among their differently watermarked copies. The parts that do

not differ between copies correspond to bits that are common to each watermark message, and are *undetectable* by the pirates. The assumption we make here was introduced in [27] and is referred to as the *marking assumption*. The collusion-secure codes described in this thesis rely on this assumption.

Where bits are undetectable, the pirates must use the only available version of that part to produce $p^{(C)}$. Where the pirates can detect a part corresponding to a particular bit, they may choose to use either the '1' or '0' state of that bit available to them to produce that part of $p^{(C)}$. The pirates may also choose to concentrate other attacks, for example, addition of noise, on the parts found to encode individual bits in order to destroy those bits. We denote a destroyed bit, which the decoder cannot decode into a valid bit, as $b_?$. Let $\mathbf{m}^{(C)}$ be the corrupted watermarked message present in the collusion-attacked pirate copy. The j^{th} bit of $\mathbf{m}^{(C)}$ is given by

$$\mathbf{m}_j^{(C)} \in \begin{cases} \{0, 1, b_?\}, & \text{if detectable} \\ \{\mathbf{m}_j^{(u_1)}\}, & \text{otherwise.} \end{cases} \tag{2.2.2}$$

If the marking assumption holds, then any bit position, j, where $\mathbf{m}_j^{(C)} = b_?$ reveals to the CP that the pirates in C had access to both the '1' and '0' version of that bit encoded in their copies. Since this gives the CP more information about the watermarked copies used by the pirates than when $\mathbf{m}_j^{(C)} = 1$ or $\mathbf{m}_j^{(C)} = 0$, producing invalid bits is a poor pirate strategy. In [27] invalid bits in $\mathbf{m}^{(C)}$ are replaced arbitrarily with '0's. In this thesis we take the same approach so we will no longer consider invalid bits as a special case.

The averaging attack described in the previous section can be regarded as a special case of the bit selection attack. To see why this is so, consider a simplified video watermarking scheme where the value of the pixel at position $x = x_0$, $y = y_0$, and $t = t_0$ encodes the watermark message bit, $\mathbf{m}_j^{(u)}$. Let $v[x_0, y_0, t_0] = \lambda_1$ encode a '1', and let $v[x_0, y_0, t_0] = \lambda_0 \neq \lambda_1$ encode a '0'. Suppose two pirates, u_1 and u_2, carry out an averaging attack. If, $\mathbf{m}_j^{(u_1)} = \mathbf{m}_j^{(u_2)} = 1$, then, according to (2.2.1), $p^{(C)}[x_0, y_0, t_0] = \lambda_1$, which encodes $\mathbf{m}_j^{(C)} = 1$. Similarly, if $\mathbf{m}_j^{(u_1)} = \mathbf{m}_j^{(u_2)} = 0$, then $p^{(C)}[x_0, y_0, t_0] = \lambda_0$, which encodes $\mathbf{m}_j^{(C)} = 0$. However, if $\mathbf{m}_j^{(u_1)} \neq \mathbf{m}_j^{(u_2)}$, then $p^{(C)}[x_0, y_0, t_0] = (\lambda_1 + \lambda_0)/2$, which does not encode a valid bit. This is the same as the marking assumption since only detectable bits can be destroyed by averaging. Therefore, when dealing with collusion attacks, we will only consider bit selection attacks.

To see why the bit selection attack is problematic, consider the following example given in [27]. Let $C = \{u_1, u_2, u_3\}$ be a set of three users who collude to produce the pirate copy, $p^{(C)}$. Let $v^{(u_1)}$, $v^{(u_2)}$, and $v^{(u_3)}$ be the uniquely watermarked videos distributed to u_1, u_2, and u_3, respectively, and let $\mathbf{m}^{(u_1)}$, $\mathbf{m}^{(u_2)}$, and $\mathbf{m}^{(u_3)}$ be the unique watermark codewords encoded into $v^{(u_1)}$, $v^{(u_2)}$, and $v^{(u_3)}$, respectively. Now suppose the pirates in C carry out a bit selection attack by selecting for each bit, $\mathbf{m}_j^{(C)}$, in $\mathbf{m}^{(C)}$, the corresponding bit found to

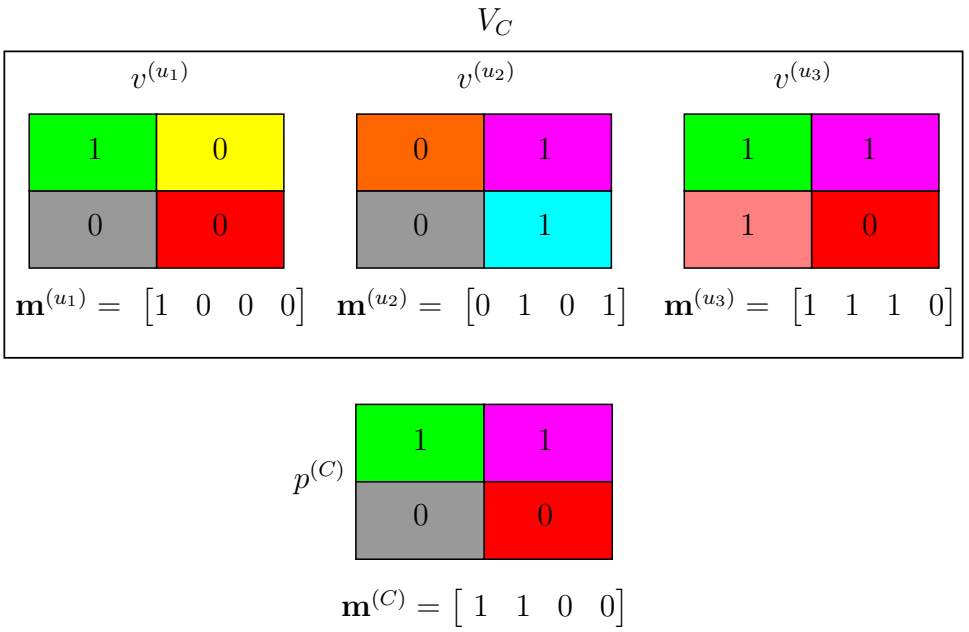


Figure 2.10: Illustration of the bit selection collusion attack. The large multi-coloured rectangles represent a frame taken from each video (at the same frame position) in the available set of copies, V_C. The small coloured regions represent parts of the frame that contain a watermark signal that encodes the bit displayed in that region. The encoded watermark messages are shown below the frames. The pirates generate the corresponding frame of the pirate copy, $p^{(C)}$, by selecting the part of each frame found to be most common among their copies. Since the bottom-left part is common to all of their copies, the pirates can only use this part for $p^{(C)}$. The resulting frame in $p^{(C)}$ could have been generated using any pair of watermarked frames, which means that no single watermarked copy can be said with certainty to have contributed to $p^{(C)}$.

occur in at least two out of the three copies available to them. That is,

$$\mathbf{m}_j^{(C)} \in \begin{cases} \mathbf{m}_j^{(u_1)}, & \text{if } \mathbf{m}_j^{(u_1)} = \mathbf{m}_j^{(u_2)} \text{ or } \mathbf{m}_j^{(u_1)} = \mathbf{m}_j^{(u_3)} \\ \mathbf{m}_j^{(u_2)}, & \text{if } \mathbf{m}_j^{(u_2)} = \mathbf{m}_j^{(u_3)} \end{cases} . \tag{2.2.3}$$

An illustration of this bit selection attack is shown in Figure 2.10. When the CP receives $p^{(C)}$, which is watermarked with $\mathbf{m}^{(C)}$, it cannot be certain that the majority strategy was actually used by the pirates. Additionally, it is possible for a group of pirates with access to any two of the three possible messages to have selected every bit of $\mathbf{m}^{(C)}$ from one of those two messages. Therefore, the CP cannot tell whether $\{u_1, u_2\}$, $\{u_1, u_2\}$, or $\{u_1, u_2\}$ created the pirate copy. Since no user is common to all of these possible pairs, no user can be incriminated with any certainty.

2.2.2 Collusion-Secure Codes

Collusion-secure binary watermarking codes were introduced by Boneh and Shaw in [27], as a solution to the bit selection collusion attack problem. They proposed a static traitor tracing scheme in which each user in U receives a copy watermarked with a unique collusion-secure codeword, constructed in such a way that a pirate copy resulting from a collusion attack carried out by C can be traced with some certainty to at least one member of C. Let $M = \{(\mathbf{m}^{(u_1)}, u_1), (\mathbf{m}^{(u_2)}, u_2) \dots (\mathbf{m}^{(u_{N_u})}, u_{N_u})\}$ be the set of ordered pairs of collusion-secure codewords (watermark messages) and the users who receive copies watermarked with those codewords. Upon finding a collusion-attacked copy, the CP decodes the corrupted watermark message and passes it to a function f_{incr} that incriminates some subset of users $\hat{C}$ as having contributed to the collusion-attacked copy, where

$$\hat{C} = f_{\text{incr}}(\mathbf{m}^{(C)}, M). \tag{2.2.4}$$

The incrimination function is probabilistic, i.e. the users in $\hat{C}$ are believed to be pirates with some probability of error, since a deterministic static traitor tracing scheme would require impractically long codes [27]. In [46] Laarhoven et al. describe the performance of a collusion-secure code construction in terms of its *soundness* and *completeness*, and we use similar definitions in this thesis. A collusion-secure code is ϵ_1-sound if the probability of accusing an innocent user of being one of the pirate collaborators is, at most, ϵ_1. That is,

$$P(\hat{C} \cap (U \setminus C) \neq \emptyset) \leq \epsilon_1. \tag{2.2.5}$$

A collusion-secure code is ϵ_2-complete if the probability of not accusing any member of C is, at most, ϵ_2. That is,

$$P(\hat{C} \cap C = \emptyset) \leq \epsilon_2. \tag{2.2.6}$$

The construction of Boneh and Shaw requires a watermark message length of

$$l = O(\hat{N}_c{}^3 \ln(N_u/\epsilon_2)) \tag{2.2.7}$$

bits for ϵ_2 completeness, where $\hat{N}_c \leq N_u$ is an upper bound on the number of pirates in C. If $N_c > \hat{N}_c$ then the scheme is no longer provably sound or complete.

2.2.3 The Codeword Construction of Tardos

In [17] Tardos introduced another construction of ϵ_1-secure, ϵ_2-complete binary collusion-secure codes for static traitor tracing, which requires a watermark message length of

$$l = O(\hat{N}_c{}^2 \ln(N_u/\epsilon_1)) \tag{2.2.8}$$

bits, where

$$\epsilon_2 = \epsilon_1^{\hat{N}_c/4}. \tag{2.2.9}$$

As with the scheme of Boneh and Shaw, $\hat{N}_c$ is an upper bound on the number of pirates in C. Tardos proved this message length to be optimal, within a constant factor, for these parameters. We describe the construction and incrimination algorithms below for $\hat{N}_c > 2$. The values of the constants that appear in the construction were selected in [17] to be $d_l = 100$, $d_\delta = 300$, and $d_\tau = 20$.

1. Set the codeword length $l = d_l \hat{N}_c^{\,2} \ln(N_u/\epsilon_1)$.

2. For each bit position, j, $1 \leq j \leq l$, select ρ'_j independently randomly from a uniform distribution over $[\delta, \frac{\pi}{2} - \delta]$, where δ is the cut-off parameter, given by
$$\delta = \arcsin\sqrt{1/d_\delta N_c}. \tag{2.2.10}$$

3. Set $\mathbf{m}_j^{(u_i)} = 1$ with probability $\rho_j = \sin^2\rho'_j$ for all $1 \leq j \leq l$, $1 \leq i \leq N_u$ and encode $\mathbf{m}^{(u_i)}$ into the copy transmitted to u_i.

4. When pirate content containing the corrupted watermark message, $\mathbf{m}^{(C)}$, is received, calculate an incrimination score for each user $u_i \in U$, using the score function,
$$s_{\text{incr}}(u_i, \mathbf{m}^{(C)}) = \sum_{j=1}^{l} \mathbf{S}_{ij}, \tag{2.2.11}$$
where $\mathbf{S}$ is a score matrix constructed as follows:
$$\mathbf{S}_{ij} = \begin{cases} +\sqrt{(1-\rho_j)/\rho_j}, & \text{if } \mathbf{m}_j^{(u_i)} = 1 \text{ and } \mathbf{m}_j^{(C)} = 1 \\ -\sqrt{\rho_j/(1-\rho_j)}, & \text{if } \mathbf{m}_j^{(u_i)} = 0 \text{ and } \mathbf{m}_j^{(C)} = 1 \end{cases} \tag{2.2.12}$$
If for any user $u_i \in U$, $s_{\text{incr}}(u_i, \mathbf{m}^{(C)}) > \tau$, where τ is a score threshold given by
$$\tau = d_\tau \ln(N_u/\epsilon_1), \tag{2.2.13}$$
then incriminate u_i as a pirate.

2.2.4 Improvements to Tardos' Codes

In [47], Škorić et al. improved the performance of Tardos' codes by using a modified, symmetric score matrix. The code construction remains the same as above, except that in step 4 the incrimination score function, s_{incr}, is replaced by the new symmetric incrimination score function, s'_{incr}, given by

$$s'_{\text{incr}}(u_i, \mathbf{m}^{(C)}) = \sum_{1}^{l} \mathbf{S}'_{ij}, \tag{2.2.14}$$

where the symmetric score matrix, $\mathbf{S}'$, is constructed as follows:

$$\mathbf{S}'_{ij} = \begin{cases} +\sqrt{(1-\rho_j)/\rho_j}, & \text{if } \mathbf{m}_j^{(u_i)} = 1 \text{ and } \mathbf{m}_j^{(C)} = 1 \\ -\sqrt{\rho_j/(1-\rho_j)}, & \text{if } \mathbf{m}_j^{(u_i)} = 0 \text{ and } \mathbf{m}_j^{(C)} = 1 \\ -\sqrt{(1-\rho_j)/\rho_j}, & \text{if } \mathbf{m}_j^{(u_i)} = 1 \text{ and } \mathbf{m}_j^{(C)} = 0 \\ +\sqrt{\rho_j/(1-\rho_j)}, & \text{if } \mathbf{m}_j^{(u_i)} = 0 \text{ and } \mathbf{m}_j^{(C)} = 0 \end{cases} . \tag{2.2.15}$$

The user $u_i \in U$ is then incriminated if $s'_{\text{incr}}(u_i, \mathbf{m}^{(C)}) > \tau$.

In [46] a constant-factor improvement was made to the length of Tardos' original codes by using the symmetric score matrix above, selecting $d_l = 23.79$, $d_\delta = 28.31$, and $d_\tau = 8.06$, and rounding the resulting code length up to the nearest integer.

2.2.5 Dynamic Tardos Codes

In [16], Laarhoven et al. modified Tardos' code construction to work as a dynamic traitor tracing scheme where the bits of each user's codeword are generated on the fly. In this dynamic version of the scheme, the incrimination score of each user is updated as each bit of the watermark message encoded in a pirate redistribution is received. When a user's incrimination score exceeds the threshold value for any value of $\hat{N}_c$ then that user is incriminated and disconnected. Their scheme was also shown to work when feedback from the pirate redistribution is delayed. Laarhoven et al. also presented a method in [16] by which optimal message lengths can be obtained for their dynamic scheme, while ensuring ϵ_1-soundness and ϵ_2-completeness for a total number of $\hat{N}_u$ users, and a maximum number of $\hat{N}_c$ collaborating pirates.

We describe the optimisation method here. For this optimisation, define γ as

$$\gamma = 1 + \frac{\ln(2)}{\ln(\hat{N}_u) - \ln(\epsilon_1)}. \tag{2.2.16}$$

Now let $g(\lambda) : (\frac{1}{2}, \infty) \to (0, \infty)$ be a function whose inverse, $g^{-1}(\phi)$, is defined as

$$g^{-1}(\phi) = \frac{e^\phi - 1 - \phi}{\phi^2}. \tag{2.2.17}$$

Since the Tardos codeword length, l, is given by $l = d_l \hat{N}_c{}^2 \ln(N_u/\epsilon_1)$, for a given $\hat{N}_c$, N_u, and ϵ_1, the optimal codeword length can be found by minimising d_l. In [16] this is done by finding parameters $\phi_1 \in (\frac{1}{2}, \infty)$, $\phi_2 \in (0, \infty)$, and $\phi_3 \in (0, \frac{2}{\pi})$ that minimise

$$d_l = \frac{\eta' \sqrt{\frac{d_\delta}{s^2 c} + d_\tau}}{\phi_3}, \tag{2.2.18}$$

where

$$\eta' = \frac{\phi_2 + \ln(2/\epsilon_2)}{\ln(\hat{N}_u/\epsilon_1)}, \tag{2.2.19}$$

$$d_\delta = \left[\frac{1}{4/\pi - 2\phi_3}\left(\sqrt{\frac{[g^{-1}(\phi_2)\phi_2]^2}{\sqrt{c}} + \frac{16}{\pi}\left(\frac{2}{\pi} - \phi_3\right)} + \frac{g^{-1}(\phi_2)\phi_2}{\sqrt{c}}\right)\right]^2, \tag{2.2.20}$$

$$d_\alpha = \max\left(\frac{\sqrt{d_\delta}}{g(\phi_1)\sqrt{\hat{N}_c}}, \frac{\phi_1}{\phi_3} + \sqrt{\left(\frac{\phi_1}{\phi_3}\right)^2 \frac{\phi_1\eta'}{\phi_3\gamma}\sqrt{\frac{d_\delta}{\phi_2^2\hat{N}_c}}}\right), \tag{2.2.21}$$

and

$$d_\tau = \frac{\phi_3 d_\alpha^2\gamma + \phi_1\eta'\sqrt{\frac{d_\delta}{\phi_2^2\hat{N}_c}}}{\phi_3 d_\alpha - \phi_1}. \tag{2.2.22}$$

Let $d_l(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2)$, $d_\delta(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2)$, and $d_\tau(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2)$ denote the values of the parameters d_l, d_t, and d_τ, respectively, that solve the above optimisation for $\hat{N}_c$ collaborating pirates, $\hat{N}_u$ users, ϵ_1-soundness, and ϵ_2-completeness. Then, for the dynamic Tardos scheme, let

$$l(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2) = d_l(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2)\hat{N}_c{}^2\ln(\hat{N}_u/\epsilon_1) \tag{2.2.23}$$

be the optimal code length,

$$\delta(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2) = \arcsin\sqrt{\frac{1}{d_\delta(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2)\hat{N}_c{}^{4/3}}} \tag{2.2.24}$$

be the optimal cut-off parameter, and

$$\tau(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2) = d_\tau(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2)\hat{N}_c\ln(\hat{N}_u/\epsilon_1) \tag{2.2.25}$$

be the optimal incrimination score threshold for $\hat{N}_c$, $\hat{N}_u$, ϵ_1, and ϵ_2. The algorithm for the dynamic scheme of Laarhoven et al. is then as follows:

1. For each bit position j, $1 \le j \le l(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2)$, select ρ'_j independently randomly from a uniform distribution over $[\delta_{\min}, \frac{\pi}{2} - \delta_{\min}]$, where $\delta_{\min} = \delta(\hat{N_{c_{\max}}}, \hat{N}_u, \epsilon_1, \epsilon_2)$.

2. For $c \in \{1, \ldots, \hat{N}_c\}$ let $\mathbf{p}(c)$ be a vector whose j^{th} element is given by

$$\mathbf{p}(c)_j = \begin{cases} 1, & \text{if } \delta(c, \hat{N}_u, \epsilon_1, \epsilon_2) \le \rho'_j \le \frac{\pi}{2} - \delta(c, \hat{N}_u, \epsilon_1, \epsilon_2) \\ 0, & \text{otherwise.} \end{cases} \tag{2.2.26}$$

3. Set $\mathbf{m}_j^{(u_i)} = 1$ with probability $\rho_j = \sin^2\rho'_j$ for all $1 \le j \le l(\hat{N}_c, \hat{N}_u, \epsilon_1, \epsilon_2)$, $1 \le i \le \hat{N}_u$ and encode $\mathbf{m}_j^{(u_i)}$ into the copy transmitted to user u_i.

4. As new pirate content is received, decode the watermark message bits contained in the content. Let $\mathbf{m}^{(C)}$ be the total watermark message received so far, and let l' be the current number of bits in $\mathbf{m}^{(C)}$. Construct the score matrix $\mathbf{S}'$ from $\mathbf{m}^{(C)}$, using (2.2.15).

5. For each user, u_i, $1 \leq i \leq N_u$, and for all $c \in \{1, \ldots, \hat{N}_c\}$ apply the dynamic incrimination score function, s_{dyn}, to $\mathbf{m}^{(C)}$, where

$$s_{\text{dyn}}(u_i, c, \mathbf{m}^{(C)}) = \sum_{1}^{l'} \mathbf{S}'_{i_j} \mathbf{p}(c)_j, \tag{2.2.27}$$

and incriminate u_i if $s_{\text{dyn}}(u_i, c, \mathbf{m}^{(C)}) > \tau(c, \hat{N}_u, \epsilon_1, \epsilon_2)$. As soon as a user is incriminated, disconnect that user and continue with the scheme.

In [16] it is shown that by setting $\delta_{\min} = 0$, and letting $\hat{N}_c \to \infty$, the above scheme can be used to trace a group of collaborating pirates of arbitrary size. As soon as, for any value of c, $s_{\text{dyn}}(u_i, c, \mathbf{m}^{(C)}) > \tau(c, \hat{N}_u, \epsilon_1, \epsilon_2)$, the user u_i is disconnected by the CP from the media distribution system. In this way, all the members of C are eventually disconnected and pirate redistribution is stopped. This dynamic scheme has some appealing properties, such as the lowest watermark message length currently found in literature, the ability to tolerate delayed feedback from the pirate redistribution, and the ability to trace any number of pirates without the need for an initial upper bound on N_c.

The dynamic scheme of Laarhoven et al. assigns a unique watermark message to each user in U. While this is not a problem for internet-based media distribution, where the server can send customised information to each client, it can be a problem for broadcast systems. In broadcast systems, data intended for any user or set of users is broadcast to all users, and some form of broadcast encryption is used to control which users can access which information [48]. In such a setting it may be advantageous to limit the amount of user-specific data that needs to be transmitted, since each unique message adds to the overall transmission bandwidth required. In the following section we describe a dynamic traitor tracing scheme that partitions U into subsets of users, and the users in each subset are always assigned the same watermark message signal. When $N_c \ll N_u$, the number of subsets required is much smaller than N_u, and so is the number of unique watermark messages required.

2.2.6 Fiat and Tassa Dynamic Scheme

The second dynamic traitor tracing scheme presented in [18] by Fiat and Tassa requires no initial upper bound on N_c, and can trace all pirates in C for any N_c. When this scheme is applied to a video broadcast, the video broadcast is partitioned, in time, into contiguous video segments. During each iteration of the traitor tracing scheme, prior to transmission of the current video segment, U is partitioned into $2N_d+1$ disjoint subsets of users, such that $U_c = \bigcup_{Q \in P_{\text{FT}}} Q$, where $P_{\text{FT}} = \{L_1, R_1, \ldots, L_{N_d}, R_{N_d}, I\}$, and $N_d \leq N_c$ is the number of pirates detected by the scheme. Each union, $L_k \cup R_k, 1 \leq k \leq N_d$, is known to contain at least one pirate, and the set I does not contain any detected pirates.

Each set of users in P_{FT} receives a differently watermarked variant of the current video segment. The CP monitors any pirate rebroadcasts of its content, and when a watermarked variant is received, the CP knows that there must be at least one pirate in the set of users to whom that particular variant was transmitted. The scheme traces each pirate by successively halving the set of users containing that pirate until a variant corresponding to a singleton set is received, and the pirate in that set is disconnected from the broadcast. We describe the Fiat and Tassa (FT) dynamic traitor tracing scheme below.

1. Set $N_d = 0$, $I = U$, and $P_{\mathrm{FT}} = \{I\}$.
2. Transmit segment variant $v^{(Q)}$ to every user in the non-empty set $Q \in P_{\mathrm{FT}}$.
3. Receive segment $v^{(P)}$ from a pirate redistribution.
4. If $v^{(P)} = v^{(I)}$, set $N_d \leftarrow N_d + 1$, split I into two sets, L_{N_d} and R_{N_d}, such that $||L_{N_d}| - |R_{N_d}|| \leq 1$, and set $P_{\mathrm{FT}} \leftarrow P_{\mathrm{FT}} \cup \{L_{N_d}, R_{N_d}\}$ and $I \leftarrow \emptyset$.
5. If $v^{(P)} = v^{(L_k)}, 1 \leq k \leq N_d$, let $Q = L_k$ and $Q' = R_k$. If $v = v^{(R_k)}, 1 \leq k \leq N_d$, let $Q = R_k$ and $Q' = L_k$.
6. Add Q' to I.
7. If $|Q| = 1$ disconnect the pirate in Q, set $P_{\mathrm{FT}} \leftarrow P_{\mathrm{FT}} \setminus \{Q, Q'\}$, set $N_d \leftarrow N_d - 1$, and renumber the sets in P_{FT} so that $P_{\mathrm{FT}} = \{L_1, R_1, \ldots, L_{N_d}, R_{N_d}, I\}$. Otherwise split Q into two sets, Q and Q', such that $||Q| - |Q'|| \leq 1$.
8. Repeat steps 2 – 7 forever.

Fiat and Tassa showed that the maximum number of unique variants required by this scheme is $2N_c + 1$. They also showed that the maximum number of iterations (repetitions of steps 2 – 7) required until all N_c pirates are disconnected is $N_c \log_2(N_u)$. However, this is only valid for a scheme where it is always possible to correctly identify every received segment variant. When digital watermarks containing probabilistic collusion-secure codes are used to generate the unique variants, there is always a chance that when a collusion-attacked pirate segment is received, it will be determined in error to have been created using some original variant. This will result in the generation of another branch of the search tree that attempts to trace a non-existent pirate. This non-existent pirate increases the requirements of the scheme.

In [28] Tassa used the collusion-secure binary watermarking code construction of Boneh and Shaw [27] to mark the segment variants transmitted in the above algorithm. It is suggested in [28] that using instead the construction of Tardos [17], the codeword length of which is a factor $O(N_c)$ smaller than than that of the construction in [27], will reduce the overall bandwidth requirements of the scheme. This is done by carrying out the codeword construction

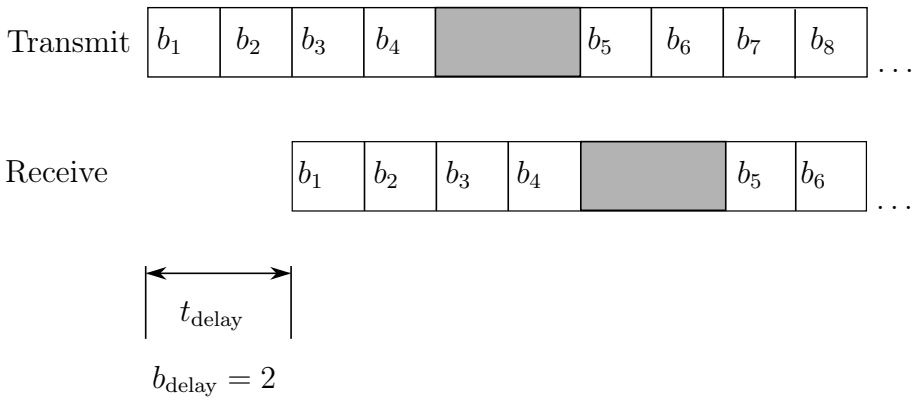


Figure 2.11: Illustration of the effect of delayed feedback from a pirate rebroadcast on the dynamic scheme of Fiat and Tassa when binary watermarks are used to produce video segment variants. The first transmitted video segment contains the watermark message b_1–b_4. The pirate version of the first segment is received by the CP t_{delay} seconds after that segment was transmitted by the CP. Since the distribution of the variants of the second segment (containing bits b_5–b_8) depends on the watermark found in the first segment, the CP transmits non-watermarked video while it waits for the first segment. At the rate at which the watermarking scheme encodes message bits, this waiting period wastes the equivalent of 2 message bits.

procedure described in Section 2.2.3 immediately before each iteration, using $\hat{N}_c = N_u = 2N_d + 1$ for some desired ϵ_1 and ϵ_2. The resulting $2N_d + 1$ codewords are then used as the watermark messages encoded into each video segment variant. When a pirate segment is received, the incrimination score function is used to identify, with certainty, which of the originally transmitted variants were used to construct the pirate segment.

2.2.7 Delayed Feedback from Pirate Rebroadcast

In the FT scheme, the variants of each video segment are only assigned to sets of users once the previously transmitted pirate segment is received by the CP. This is acceptable if the pirate content is rebroadcast simultaneously with the original broadcast. However, if there is a delay between transmitting a segment and receiving the pirated version of that segment, then the traitor tracing scheme must wait for the received segment before updating P. Since the video broadcast must be continuous, this means that either non-watermarked content, or content watermarked with redundant information would have to be broadcast while the traitor tracing scheme waits for feedback from the pirate rebroadcast.

We assume that the digital watermarking scheme embeds watermark messages into the video segments at a constant rate of r_{wm} bits per second of video broadcast. We also assume that the pirate rebroadcast is delayed by a constant t_{delay} seconds, with respect to the original broadcast, and that the

CP has access to the delayed pirate rebroadcast. The number of watermark message bits, b_{delay}, that can be transmitted during t_{delay} is then given by $b_{\text{delay}} = r_{\text{wm}} t_{\text{delay}}$. For the FT scheme, using Tardos codes to mark the segment variants, the equivalent of b_{delay} bits of digital watermark bits are wasted during every iteration while the scheme waits for feedback. Figure 2.11 illustrates this effect.

Chapter 3

System Design

We will now describe how we apply the techniques reviewed in the previous chapter to the design of a digital video watermarking scheme with dynamic traitor tracing. We begin by describing the design of a digital watermarking algorithm for encoding arbitrary binary messages into uncompressed digital video. We then go on to describe the dynamic traitor tracing scheme used to construct the watermark message signals assigned to each user in such a way that the entire watermarking scheme is robust against collusion attacks.

3.1 The Watermarking Scheme

The digital video watermark modulation scheme presented here consists of an encoder and decoder, as shown in Figure 1.1 on page 3. Our implementation is based on the STDM technique, introduced by Chen and Wornell [39], and described in Section 2.1.6 on page 14. This section describes the algorithms implemented by the encoder and decoder.

3.1.1 Video Blocks

Prior to watermark encoding or decoding, the video signal is partitioned along the time axis into K contiguous blocks, each consisting of T_b frames, where $K = \lfloor T/T_b \rfloor$. We call this partitioning function f_b. The k^{th} block of a video $v[x, y, t]$ is given by

$$f_b\left(v[x, y, t], k\right) = v[x, y, t + (k-1)T_b]\left[u(0) - u(t - T_b)\right], \tag{3.1.1}$$

where $k \in \{1, \ldots, K\}$, and $u(t)$ is a unit step function defined as

$$u(t) = \begin{cases} 1, & t \geq 0 \\ 0, & \text{otherwise} \end{cases}. \tag{3.1.2}$$

Figure 3.1 illustrates the partitioning of a video signal into blocks. In each block we encode N_b bits of the watermark message signal, $\mathbf{m}^{(u)}$. This segmentation ensures that message bits can be recovered from snippets of watermarked

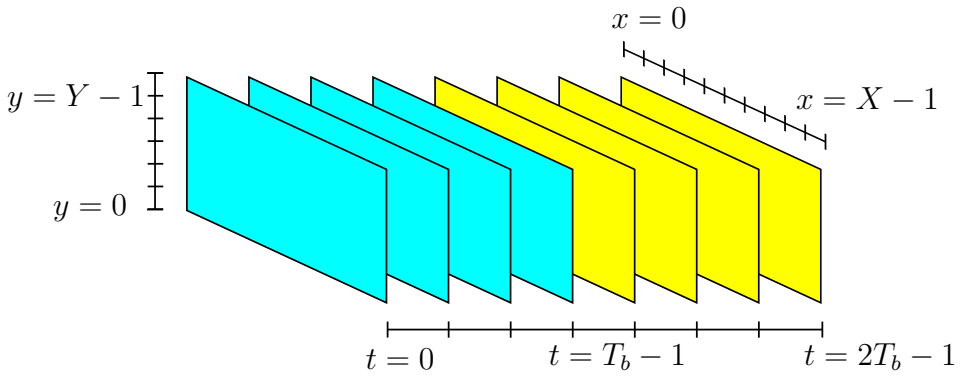


Figure 3.1: The digital video signal, v, partitioned along the time axis by the partitioning function, f_b, into contiguous blocks, each containing $T_b = 4$ video frames. The blue frames make up the first block, $f_b(v, 1)$, and the yellow frames make up the second block, $f_b(v, 2)$.

video that still contain entire blocks. We denote the i^{th} message bit encoded into the k^{th} video block as b_{k_i}, where $i \in \{1, \ldots, N_b\}$, and $k \in \{1, \ldots, K\}$.

3.1.2 Chip Signals

The "transform" step of STDM encoding and decoding involves multiplying the host signal with a pseudo-noise chip signal. In our scheme, the i^{th} bit, b_{k_i}, of the k^{th} video block is encoded and decoded using a pseudorandomly generated chip signal $s_{k_i}[x', y', t']$, where $x' \in \{0, \ldots, X_s - 1\}$, $y' \in \{0, \ldots, Y_s - 1\}$, and $t' \in \{0, \ldots, T_s - 1\}$. We choose $X_s \ll X$, $Y_s \ll Y$, and $T_s \ll T_b$ in order to give the chip signal a significantly lower spatial and temporal resolution than the host signal. This is intended to allow the decoder to tolerate minor temporal and spatial misalignment of the video signal with respect to $s_{k_i}[x', y', t']$.

We generate $s_{k_i}[x', y', t']$ as follows. Let Ω_k be the set of all possible coordinates, $[x'', y'', t'']$, where $x'' \in \{0, \ldots, X_s - 1\}$, $y'' \in \{0, \ldots, Y_s - 1\}$, and $t'' \in \{0, \ldots, T_s - 1\}$. We randomly partition Ω_k into $N_b + 1$ disjoint subsets, such that

$$\Omega_k = \bigcup_{i=1}^{N_b+1} \Omega_{k_i}. \tag{3.1.3}$$

For $i \in \{1, \ldots, N_b\}$, $|\Omega_{k_i}| = N_s$, where

$$N_s = \text{weight}(s_{k_i}) = \lfloor X_s Y_s T_s / N_b \rfloor \tag{3.1.4}$$

is the number of coordinates corresponding to non-zero values of $s_{k_i}[x', y', t']$. $\Omega_{k_{N_b+1}}$ is the set of coordinates not assigned to any chip signal when $X_s Y_s T_s$ is not integer divisible by N_b, such that $|\Omega_{k_{N_b+1}}| = X_s Y_s T_s - N_b \lfloor X_s Y_s T_s / N_b \rfloor$. The chip signal, $s_{k_i}[x', y', t']$, is given by

$$s_{k_i}[x', y', t'] = \sum_{[x'', y'', t''] \in \Omega_{k_i}} s'_k[x', y', t'] \delta[x' - x'', y' - y'', t' - t''], \tag{3.1.5}$$

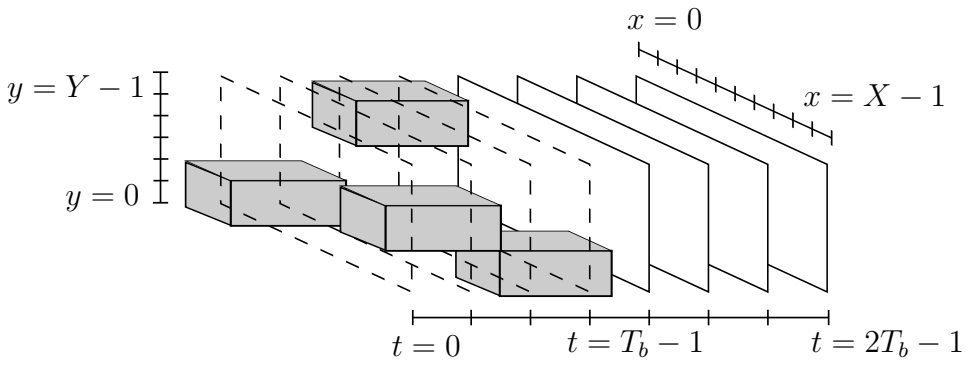


Figure 3.2: Example of a chip signal, $s_{1_2}[x', y', t']$, that is used to encode the second message bit in the first video block. The shaded regions correspond to the locations where $s_{1_2}[x', y', t']$ is non-zero. The low resolution chip signal is shown here overlaid onto the high resolution host signal, $v[x, y, t]$, with the frames of the first block of v drawn using dashed lines. In this example, the host video resolution is given by $X = 10$, $Y = 6$, $T_b = 4$, and the chip signal resolution is given by $X_s = 5$, $Y_s = 3$, $T_s = 2$. Therefore, each non-zero region of s_{1_2} covers 2×2 pixels $\times 2$ frames of the host signal, and is referred to as a cell.

where $i \in \{1, \ldots, N_b\}$, $\delta[x' - x'_0, y' - y'_0, t' - t'_0]$ is the Kronecker delta and $s'_k[x', y', t']$ is a function generated by a uniform random selection from the set $\{-1, 1\}$ for each coordinate in Ω_k. Figure 3.2 shows an example of a single chip signal within a video block.

3.1.3 Downsampling and Spread-Transformation

Before a video signal block can be multiplied with a chip signal, it must be downsampled to the spatial and temporal resolution of the chip signal. We represent the downsampling operation as a transform, $\mathcal{D}\{v\}$, from the video block, $v_k[x, y, t] = f_b(v, k)$, consisting of T_b $X \times Y$-pixel frames to the low resolution video signal, $\tilde{v}_k[x', y', t']$, consisting of T_s $X_s \times Y_s$-pixel frames, such that

$$\tilde{v}_k[x', y', t'] = \mathcal{D}\{v_k[x, y, t]\}, \tag{3.1.6}$$

where $x' \in \{0, \ldots, X_s - 1\}$, $y' \in \{0, \ldots, Y_s - 1\}$, and $t' \in \{0, \ldots, T_s - 1\}$. We calculate $D\{v_k[x, y, t]\}$ as follows:

$$\mathcal{D}\{v_k[x, y, t]\} = \frac{1}{X_c Y_c T_c} \sum_{i'=0}^{X_c-1} \sum_{j'=0}^{Y_c-1} \sum_{k'=0}^{T_c-1} v_k\big[x' X_c + i', y' Y_c + j', t' T_c + k'\big], \tag{3.1.7}$$

where $X_c = \lfloor X/X_s \rfloor$, $Y_c = \lfloor Y/Y_s \rfloor$, and $T_c = \lfloor T_b/T_s \rfloor$ are the width and height, in pixels, and length, in frames, respectively, of the region of the video signal that affects a single value of the downsampled version of that video. We refer to such a region as a *cell*. We also define a new spread-transformation function,

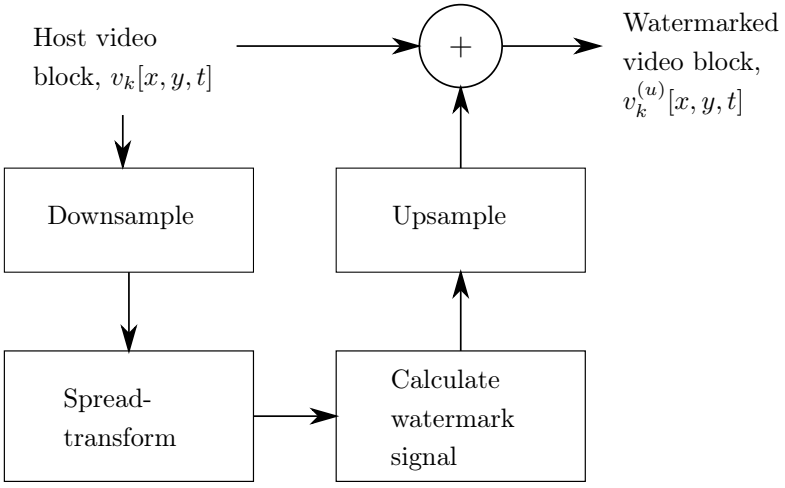


Figure 3.3: Diagram of the digital watermark encoder. The k^{th} block of the host video, $v_k[x, y, t]$, is first downsampled to the resolution of the chip signals. Finally the watermark signal is upsampled to the resolution of the host, and added to it to give the watermarked video block, $v_k^{(u)}[x, y, t]$.

$\tilde{f}_p$, which projects the downsampled version of a video block, $v_k[x, y, t]$, onto a given chip signal, $s[x', y', t']$, as

$$\tilde{f}_p(v_k[x, y, t], s[x', y', t']) = \sum_{x'=0}^{X_s-1} \sum_{y'=0}^{Y_s-1} \sum_{t'=0}^{T_s-1} s[x', y', t'] \mathcal{D}\{v_k[x, y, t]\}. \tag{3.1.8}$$

3.1.4 The Encoder

Figure 3.3 shows a diagram of the watermark encoder. The host video, $v[x, y, t]$, is first partitioned into blocks using (3.1.1). The k^{th} block of v is $v_k = f_b(v, k)$. For each $i \in \{1, \ldots, N_b\}$, the encoder calculates $\tilde{f}_p(v_k[x, y, t], s_{k_i}[x', y', t'])$, using (3.1.8), and then finds the nearest DM value that encodes the bit b_{k_i}, given by

$$\tilde{q}_{\text{STDM}}(v_k, b_{k_i}, \Delta, s_{k_i}) = q_{\text{DM}}\left(\tilde{f}_p(v_k[x, y, t], s_{k_i}[x', y', t']), b_{k_i}, \Delta\right), \tag{3.1.9}$$

where q_{DM} is defined in (2.1.2). The quantiser step size, Δ, used for encoding and decoding, depends on the parameters of the chip signal, s_{k_i}, and its value is calculated later in (3.1.15).

The next step in the STDM encoding procedure is to calculate the low resolution watermark signal, $w_{k_i}^{(u)}[x', y', t']$, for each bit b_{k_i}, such that

$$\tilde{f}_p(v_k + w_{k_i}^{(u)}, s_{k_i}) = \tilde{q}_{\text{STDM}}(v_k, b_{k_i}, \Delta, s_{k_i}). \tag{3.1.10}$$

Let z_{k_i} be the scalar difference between the DM value that encodes b_{k_i} and the value of the projection of v_k onto s_{k_i}, given by

$$z_{k_i} = \tilde{q}_{\text{STDM}}(v_k, b_{k_i}, \Delta, s_{k_i}) - \tilde{f}_p(v_k, s_{k_i}). \tag{3.1.11}$$

Let $\tilde{w}_{k_i}^{(u)}[x', y', t']$ denote the watermark signal, $w_{k_i}^{(u)}[x, y, t]$, downsampled to the resolution on s_{k_i}, so that

$$\tilde{w}_{k_i}^{(u)}[x', y', t'] = \mathcal{D}\left\{w_{k_i}^{(u)}[x, y, t]\right\}. \tag{3.1.12}$$

From (2.1.7), $\tilde{w}_{k_i}^{(u)}$ is calculated as

$$\tilde{w}_{k_i}^{(u)} = \frac{z_{k_i} s_{k_i}}{\text{weight}(s_{k_i})}. \tag{3.1.13}$$

In our implementation, we limit $\tilde{w}_{k_i}^{(u)}[x', y', t']$, so that when the magnitude of $\tilde{w}_{k_i}^{(u)}[x', y', t']$ is a maximum, the maximum magnitude of the change in the pixel values of the host video is equal to 1. This condition occurs when $\tilde{w}_{k_i}^{(u)}[x', y', t'] = s_{k_i}[x', y', t']$. That is,

$$\max\left(\tilde{w}_{k_i}^{(u)}\right) = \frac{\max(z_{k_i}) s_{k_i}}{N_s} = s_{k_i}, \tag{3.1.14}$$

where $N_s = \text{weight}(s_{k_i})$, and $\max(z_{k_i})$ is the maximum distance between the DM value that encodes b_{k_i} and the projection of the host signal onto s_{k_i}. In Figure 2.4 on page 13 we see that $\max(z_{k_i}) = \Delta/2$. Substituting into (3.1.14), we get

$$\Delta = 2N_s. \tag{3.1.15}$$

We choose this value of Δ to be used by the encoder and decoder.

Using (3.1.13) would result in values of $w_{k_i}^{(u)}$ that are not necessarily integral. Since the pixel values of the digital video signal are integers, we want the watermark signal to also be integral. We do this by randomly selecting a subset Ω'_{k_i} of Ω_{k_i} such that $|\Omega'_{k_i}| = \text{round}(z_{k_i})$, and calculating $\tilde{w}_{k_i}^{(u)}$ as

$$\tilde{w}_{k_i}^{(u)}[x', y', t'] = \text{sgn}(z_{k_i}) \sum_{[x'', y'', t''] \in \Omega'_{k_i}} s'[x', y', t'] \delta[x' - x'', y' - y'', t' - t'']. \tag{3.1.16}$$

The total low resolution watermark signal for block v_k, $\tilde{w}_k^{(u)}[x', y', t']$, is then given by

$$\tilde{w}_k^{(u)}[x', y', t'] = \sum_{i=1}^{N_b} \tilde{w}_{k_i}^{(u)}[x', y', t']. \tag{3.1.17}$$

This signal is then upsampled to the resolution of the host video signal, $v[x, y, t]$, by simply repeating the value at each coordinate of $\tilde{w}_k^{(u)}[x', y', t']$ over its corresponding cell to give

$$w_k^{(u)}[x, y, t] = \tilde{w}_k^{(u)}\left[\lfloor x/X_c \rfloor, \lfloor y/Y_c \rfloor, \lfloor t/T_c \rfloor\right], \tag{3.1.18}$$

for $x \in \{0, \ldots, X-1\}$ and $y \in \{0, \ldots, Y-1\}$, and $t \in \{0, \ldots, T_b-1\}$. Finally, the encoder generates the watermarked video block, $v_k^{(u)}[x, y, t]$, as

$$v_k^{(u)}[x, y, t] = v_k[x, y, t] + w_k^{(u)}[x, y, t]. \tag{3.1.19}$$

3.1.5 Attacks on the Watermark

A pirate wishing to distribute the watermarked video, $v^{(u)}[x, y, t]$, may attempt to attack the video with the intention of preventing the decoder from finding a valid watermark signal. We consider this attack to be a combination of amplitude scaling (gain), frame de-synchronisation, and noise addition. The pirate version of $v^{(u)}[x, y, t]$ is then given by

$$p^{(u)}[x, y, t] = \alpha(t) f_r\big(v^{(u)}[x, y, t]\big) + n[x, y, t], \tag{3.1.20}$$

where $\alpha(t)$ is the applied pixel value scaling factor, which may vary over the duration of the video signal, $n[x, y, t]$ is an added noise signal, and f_r is a function that modifies the frame structure of the watermarked video. We assume that quick variations of $\alpha(t)$ would result in unacceptable flicker and degradation of the pirate video, so that $\alpha(t)$ varies slowly with time. We further assume that $\alpha(t)$ is approximately constant over the duration of a video block, so that over the k^{th} block, $\alpha(t) \approx \alpha_k$ for all $t \in \{(k-1)T_b, \ldots, kT_b - 1\}$.

The frame modifying function, f_r, can involve frame rate conversion and deletion of frames. The attack may also involve changing the location of the watermarked video blocks within the video. Deleting frames that occur before a block will cause that block to start earlier in the attacked video than in the original watermarked video. Adding or removing entire segments of contiguous frames by, for example, removing title sequences or adding segments of "black", could also cause significant changes to the locations of blocks. In some cases, a pirate may even change the order of scenes within a movie, which would reorder some video blocks. If a significant part of a single block is removed or cut off by reordering, then that block will likely not be correctly decodable from the video. For this reason, message bits should be embedded with enough redundancy between blocks so that the message can still be recovered when some blocks are lost.

3.1.6 Video Block Synchronisation

Figure 3.4 shows a diagram of the digital watermark decoder. When the pirate version of $v^{(u)}$, $p^{(u)}$, is received, the decoder must identify the block, $p_k^{(u)}[x, y, t]$, in $p^{(u)}$ corresponding to the block, $v_k^{(u)}[x, y, t]$, in $v^{(u)}$ for each $k \in \{1, \ldots, K\}$. The decoder is perfectly synchronised with the encoder for the k^{th} video block when each pixel in $p_k^{(u)}[x, y, t]$ corresponds to the pixel in $v_k^{(u)}[x, y, t]$ at the same coordinates. That is,

$$p_{k_{\text{sync}}}^{(u)}[x, y, t] = \alpha_k v_k^{(u)}[x, y, t] + n_k[x, y, t], \tag{3.1.21}$$

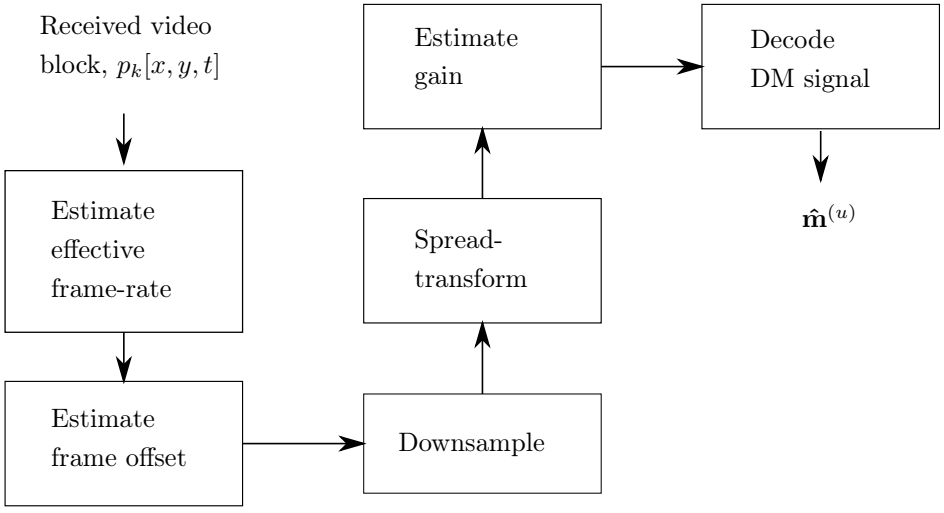


Figure 3.4: Diagram of the digital watermark decoder.

where $n_k[x, y, t]$ is the noise signal over block k, and

$$v_k^{(u)}[x, y, t] = f_b(v^{(u)}, k) \tag{3.1.22}$$

is the k^{th} watermarked video block.

In order to achieve temporal synchronisation, the decoder must locate the part of the pirate video signal corresponding to each block of the watermarked video, and attempt to reverse any frame modifications made to the block by f_r. We approximate $p_k^{(u)}$ as

$$p_k^{(u)}[x, y, t] \approx \alpha_k f_b\left(v_k^{(u)}[x, y, r_k t + \theta_k], k\right) + n_k[x, y, t], \tag{3.1.23}$$

where r_k is an approximate factor by which the frame rate of $v_k^{(u)}[x, y, t]$ has been sped up or slowed down, and θ_k is the frame offset of $f_r(v_k^{(u)}[x, y, t])$ from $v_k^{(u)}[x, y, t]$. From (3.1.21), (3.1.22), and (3.1.23) we can see that for $r_k = 1$ and $\theta_k = 0$, p_k and $v_k^{(u)}$ are synchronised. Therefore, in order to achieve synchronisation when $r_k \neq 1$ and $\theta_k \neq 0$, the decoder must estimate r_k and θ_k, and use these estimates to generate an estimate of $p_{k_{\text{sync}}}^{(u)}$, $\hat{p}_{k_{\text{sync}}}^{(u)}$, given by

$$\hat{p}_{k_{\text{sync}}}^{(u)}[x, y, t] = f_b\left\{p[x, y, (t - \hat{\theta}_k)/\hat{r}_k], k\right\}, \tag{3.1.24}$$

where $\hat{r}_k$ and $\hat{\theta}_k$ are the decoder estimates of r_k and θ_k, respectively. In order to simplify finding $\hat{r}_k$, we fix the frame rate at which the simple decoder decodes watermarks to be r_{dec} frames per second (FPS). The decoder converts any video it receives to r_{dec} FPS before decoding. Therefore, in order to synchronise with the decoder, the encoder must embed watermarks at r_{dec} FPS. This does not,

however, limit the watermarking scheme to videos with a frame rate of r_{dec} FPS, since the frame rate of the watermarked videos may be converted by the CP from r_{dec} to the desired frame rate. This conversion would simply be reversed by the decoder.

If the frame rate of a watermarked video is simply converted to $r_{\text{conv}} \neq r_{\text{dec}}$ by a pirate using some video conversion tool, then the conversion is reversed by the decoder when the frame rate of $p^{(u)}$ is converted back to r_{dec}, and therefore r_k will simply be equal to 1. However, if a number of frames are deleted from a particular block of watermarked video then the frame rate over that block is effectively reduced since there are now fewer frames that represent that block. The ratio of the number of frames of block k present in the pirate video to the number of frames in $v_k^{(u)}$ then gives an approximation of r_k.

The decoder finds $\hat{r}_k$ and $\hat{\theta}_k$ for p_k by sweeping the value of $\hat{r}_k$ over the set R, and sweeping the value of $\hat{\theta}_k$ over the set Θ, where

$$\Theta = \{1, 2 \ldots, T - T_b\}, \tag{3.1.25}$$

and the choice of values in R is an implementation detail that is discussed in Chapter 4. For each value of $\hat{r}'_k \in R$ and $\hat{\theta}'_k \in \Theta$, the decoder generates the scalar DM value, $\hat{z}_{k_i}(\hat{r}'_k, \hat{\theta}'_k)$, corresponding to each decoded bit $\hat{b}_{k_i}$, given by

$$\hat{z}_{k_i}(\hat{r}'_k, \hat{\theta}'_k) = \tilde{f}_p \left(f_b \left\{ p^{(u)} \left[x, y, (t - \hat{\theta}'_k)/\hat{r}'_k \right], k \right\}, s_{k_i} \right), \tag{3.1.26}$$

where $i \in \{1, \ldots, N_b\}$, and $\tilde{f}_p$ is defined in (3.1.8). The decoder then carries out the gain estimation algorithm, described in the following section, using $\hat{z}_{k_i}(\hat{r}'_k, \hat{\theta}'_k)$ for all $i \in \{1, \ldots, N_b\}$. The gain estimation algorithm produces a likelihood score for each combination of $\hat{\alpha}'_k$, $\hat{r}'_k$, and $\hat{\theta}'_k$ tested. The maximum scoring combination is then used for the decoder-estimated values.

3.1.7 Gain Estimation

We implement a similar gain estimation method to the one presented by Eggers et al. in [44]. The decoder generates the histogram, $h_k[\zeta]$, of $\hat{z}_{k_i}(\hat{r}'_k, \hat{\theta}'_k)$ for every $i \in \{1, \ldots, N_b\}$, using a bin size of Δ_{bin} given by

$$\Delta_{\text{bin}} = \frac{\Delta}{f_\Delta}, \tag{3.1.27}$$

where f_Δ is a constant factor that determines the resolution of the histogram relative to the quantiser step size used by the encoder.

As shown in Figure 2.9 on page 18, if the noise signal is small, $h_k[\zeta]$ should have peaks centred on the bins containing $\tilde{q}_{\text{STDM}}(\alpha_k v_k, b_{k_i}, \alpha_k \Delta, s_{k_i})$ for $i \in \{1, \ldots, N_b\}$. These DM modulated magnitudes take on values in $\{k\alpha_k\Delta/2 + \alpha_k\Delta/4, k \in \mathbb{Z}\}$, which means that $h_k[\zeta]$ appears somewhat periodic with period

$\alpha_k\Delta/2$. The histogram should therefore have a large frequency component at $f_h = 2/(\alpha_k\Delta)$. The decoder estimate of f_h, $\hat{f}_h$, is calculated as

$$\hat{f}_h = \arg\max_f \left(|H_k[f]|\right) f_s, \tag{3.1.28}$$

where $H_k[f]$ is the discrete Fourier transform (DFT) of the histogram $h_k[\zeta]$, defined as

$$H_k[f] = \sum_{\zeta=0}^{N_b-1} h_k[\zeta] e^{-j2\pi f\zeta/N_b}, \tag{3.1.29}$$

and f_s is the sampling frequency of $h[\zeta]$, given by

$$f_s = \frac{1}{\Delta_{\text{bin}}}. \tag{3.1.30}$$

The decoder can then determine $\hat{\alpha}_k$, an estimate of α_k, by comparing $\hat{f}_h$ to the expected quantiser step size, Δ:

$$\hat{\alpha}_k = \frac{2}{\Delta\hat{f}_h}. \tag{3.1.31}$$

The decoder calculates a likelihood score between 0 and 1 as

$$\beta_k = \max(|H_k[f]|)/N_b. \tag{3.1.32}$$

A large value of β_k results when $h_k[z]$ has a large component with period $\alpha_k\Delta/2$, indicating that $f_b(p^{(u)}[x, y, (t-\hat{\theta}'_k)/\hat{r}'_k], k)$ is likely to have been STDM modulated with a step size of $\hat{\alpha}_k\Delta$. Therefore, the decoder can use β_k as a measure confidence that $f_b(p^{(u)}[x, y, (t-\hat{\theta}'_k)/\hat{r}'_k], k)$ corresponds to the k^{th} block of the watermarked video, $v^{(u)}$, and that $\hat{\alpha}_k = \alpha_k$.

Let $\hat{\alpha}_k(\hat{r}'_k, \hat{\theta}'_k)$ be the estimated gain and $\beta_k(\hat{r}'_k, \hat{\theta}'_k)$ be the corresponding likelihood score, calculated as above, for the frame rate scaling factor, $\hat{r}'_k$, and block offset, $\hat{\theta}'_k$. The final decoder estimated values of r_k, θ_k, and α_k are then given by

$$\hat{r}_k = \arg\max_{\hat{r}'_k \in R} \left[\beta_k(\hat{r}'_k, \hat{\theta}'_k)\right], \tag{3.1.33}$$

$$\hat{\theta}_k = \arg\max_{\hat{\theta}'_k \in \Theta} \left[\beta_k(\hat{r}'_k, \hat{\theta}'_k)\right], \tag{3.1.34}$$

$$\hat{\alpha}_k = \hat{\alpha}_k(\hat{r}_k, \hat{\theta}_k), \tag{3.1.35}$$

respectively. The decoder then calculates the decoder synchronised block, $\hat{p}^{(u)}_{k_{\text{sync}}}[x, y, t]$, from these estimated parameters using (3.1.24).

3.1.8 Watermark Message Decoding

Once the decoder has estimated the temporal synchronisation and amplitude scaling parameters, the watermark message bits can be decoded from the received video signal. For the k^{th} video block, for each $i \in \{1, \ldots, N_b\}$, the decoder determines the decoded bit, $\hat{b}_{k_i}$, by applying the decoding rule in (2.1.10) to $\hat{p}^{(u)}_{k_{\text{sync}}}[x, y, t]$, using a quantiser step size scaled by $\hat{\alpha}$, to give

$$\hat{b_{k_i}} = \underset{b' \in \{0,1\}}{\arg\min} \left| \tilde{f}_p \left(\hat{p}^{(u)}_{k_{\text{sync}}}[x, y, t], s_{k_i} \right) - \tilde{q}_{\text{STDM}} \left(\hat{p}^{(u)}_{k_{\text{sync}}}, b', \hat{\alpha}_k \Delta, s_{k_i} \right) \right|, \quad (3.1.36)$$

where $\tilde{q}_{\text{STDM}}$ is defined in (3.1.9). The bits decoded from each watermarked block are then used to construct the message signal, $\tilde{\mathbf{m}}^{(u)}$, by reversing the method used by the encoder to distribute the bits of $\mathbf{m}^{(u)}$ among the video blocks.

3.2 The Dynamic Traitor Tracing Scheme

In this section we describe the design of a dynamic traitor tracing algorithm that can be used to make our digital watermarking scheme (or other suitable digital watermarking schemes) robust against collusion attacks. Our scheme is designed to be used in broadcast systems where feedback from the pirate redistribution is available to the CP. Like the FT scheme, our dynamic tracing algorithm is executed in iterations, and during each iteration a segment of video content is broadcast. Our scheme also partitions the users in U into subsets, and each subset is assigned a differently watermarked variant of each segment. Unlike the original FT scheme [18], described in Section 2.2.6, our scheme is designed in such a way that video segments are marked before broadcast, regardless of whether previous marked segments have been received. The aim of this approach is to continue to trace pirates during the delay between the transmission of marked segments and the receipt of illegally rebroadcast segments. When feedback from the pirate rebroadcast is delayed, we will show that this reduces the time needed to disconnect all pirates.

3.2.1 Delay Tolerant Search Tree

In our scheme we trace colluding pirates using a search tree similar to the one used by Fiat and Tassa in [28], consisting of N_l levels. Again, let U be the set of users to which the CP distributes content. Let D be the set of users that have been disconnected by the CP. Within the k^{th} tree level with $1 \leq k \leq N_l$, let W_k be the set of disjoint user sets $W \subset U$ for all $W \in W_k$ suspected of containing pirates, and let $W'_k \subseteq U$ be the set of users that did not contain any detected pirates at the time that level k was generated. Let G_p be the set of disjoint user sets, where each set, $G \in G_p$, is known to contain at least one

pirate. We refer to each set in G_p as being *guilty*. Let I be the set of users that does not contain a detected pirate. We refer to I as being *innocent*, however this is merely an assumption made until the existence of a pirate in this set is determined. Let S_p be a set of user sets suspected of containing pirates, such that for each set of users $G \in G_p$ there are a number of disjoint sets of users in S_p whose union is G. The tree is initialised by setting $I = U$, $G_p = \emptyset$, $S_p = \emptyset$, $D = \emptyset$, $W_0 = \emptyset$, $W' = U$, and $l = 0$.

Algorithm 1 The tree level generating function.

function NEWLEVEL
$N_l \leftarrow N_l + 1$
$W'_{N_l} \leftarrow I$
$W_{N_l} \leftarrow \cup\{\text{SPLIT}(Q)\ \forall\ Q \in S_p\}$
$S \leftarrow W_{N_l}$
end function

Algorithm 1 shows the function used to generate a new level of the search tree. A new level is generated after each video segment has been transmitted. When a new level is generated, the current tree level, N_l, is incremented, and each set of suspect users in S_p is split into two new, approximately equally sized subsets, using the splitting function shown in Algorithm 2. Note that the only requirement of the splitting function is that the size of each of the resulting disjoint subsets is half that of the original set. Which of the original users end up in either set is arbitrary. The set of resulting user sets is denoted W_{N_l}. Each user set in W_{N_l} is assigned a unique segment variant. The users in the innocent set, I, are added to W'_{N_l}. If $W'_{N_l} \neq \emptyset$ then W'_{N_l} also receives a unique segment variant. Section 3.2.3 discusses the watermarking of video segment variants.

Algorithm 2 The user set splitting function.

function SPLIT(S)
return $\{S_1, S_2 \subset S\ :\ |S_1| - |S_2| \leq 1, S \setminus S_2 = S_1\}$
end function

When feedback from a pirate redistribution indicates that at least one user belonging to a particular set $P \in W_k \cup \{W'_k\}$ is contributing to the collusion-attacked pirate copy, the function described in Algorithm 3 is applied to the search tree. Section 3.2.3 discusses how the feedback is used to incriminate P as containing at least one pirate. The basic operation of Algorithm 3 can be described as follows.

If P is a subset of one of the guilty sets $G \in G_p$, then the algorithm assumes that all other users in G are innocent, removes $G \setminus P$ from G, as well as from any suspect sets in S_p, and adds them to the innocent set, I. Any empty

set in S_p is removed from S_p. If all users in P were assumed innocent, i.e. $P \subseteq I$, then a new suspect set, containing all users in P, is added to S_p. If $|P| = 1$ then the single user in P is identified as a pirate, disconnected from the broadcast, removed from any sets that contain that user, and added to D.

Algorithm 3 The filter function.

function FILTER(P)
  for all $G \in G_p$ **do** ▷ Check if P is a subset of a guilty set.
    if $P \subseteq G$ **then** ▷ If so,
      $I \leftarrow I \cup G \setminus P$ ▷ assume the rest of the guilty set is innocent.
      if $|P| = 1$ **then** ▷ If there is only one user in P,
        $D \leftarrow D \cup P$ ▷ disconnect user in P.
        $S_p \leftarrow \{Q : Q = (S \setminus G) \neq \emptyset \ \forall S \in S_p\}$ ▷ Remove G ▷ from suspects.
        $G_p \leftarrow G_p \setminus \{P\}$ ▷ Remove P from guilty sets.
      else
        $S_p \leftarrow \{Q : Q = S \setminus (G \setminus P) \neq \emptyset \ \forall S \in S_p\}$ ▷ Remove $G \setminus P$ ▷ from suspects.
        $G_p \leftarrow (G_p \setminus G) \cup \{P\}$ ▷ Add P to guilty sets.
      end if
    else
      $S_p \leftarrow \{S : S \in S_p, S \not\subseteq G\} \cup \{G\}$ ▷ Recombine subsets of G.
    end if
  end for
  if $P \subseteq I$ **then** ▷ If P incriminates "innocent" users,
    $I \leftarrow I \setminus P$ ▷ remove P from innocent set.
    if $|P| = 1$ **then** ▷ If there is only on user in P,
      $D \leftarrow D \cup P$ ▷ disconnect user in P.
    else
      $G_p \leftarrow G_p \cup \{P\}$ ▷ Add P to guilty sets.
      $S_p \leftarrow S_p \cup \{P\}$ ▷ Add P to suspect sets.
    end if
  end if
end function

Figures 3.5 and 3.6 show an example of the use of Algorithms 1 and 3 to trace two colluding pirates, u_2 and u_7, amongst eight users. The states of the sets, I, G_p, S_p, W_{N_l}, W'_{N_l}, and D are shown after each algorithm is executed. The argument to the filtering function is the set of users incriminated by the receipt of a watermarked variant that was transmitted to that set of users. After the two colluding pirates have been disconnected, all other users are returned to the innocent set, I.

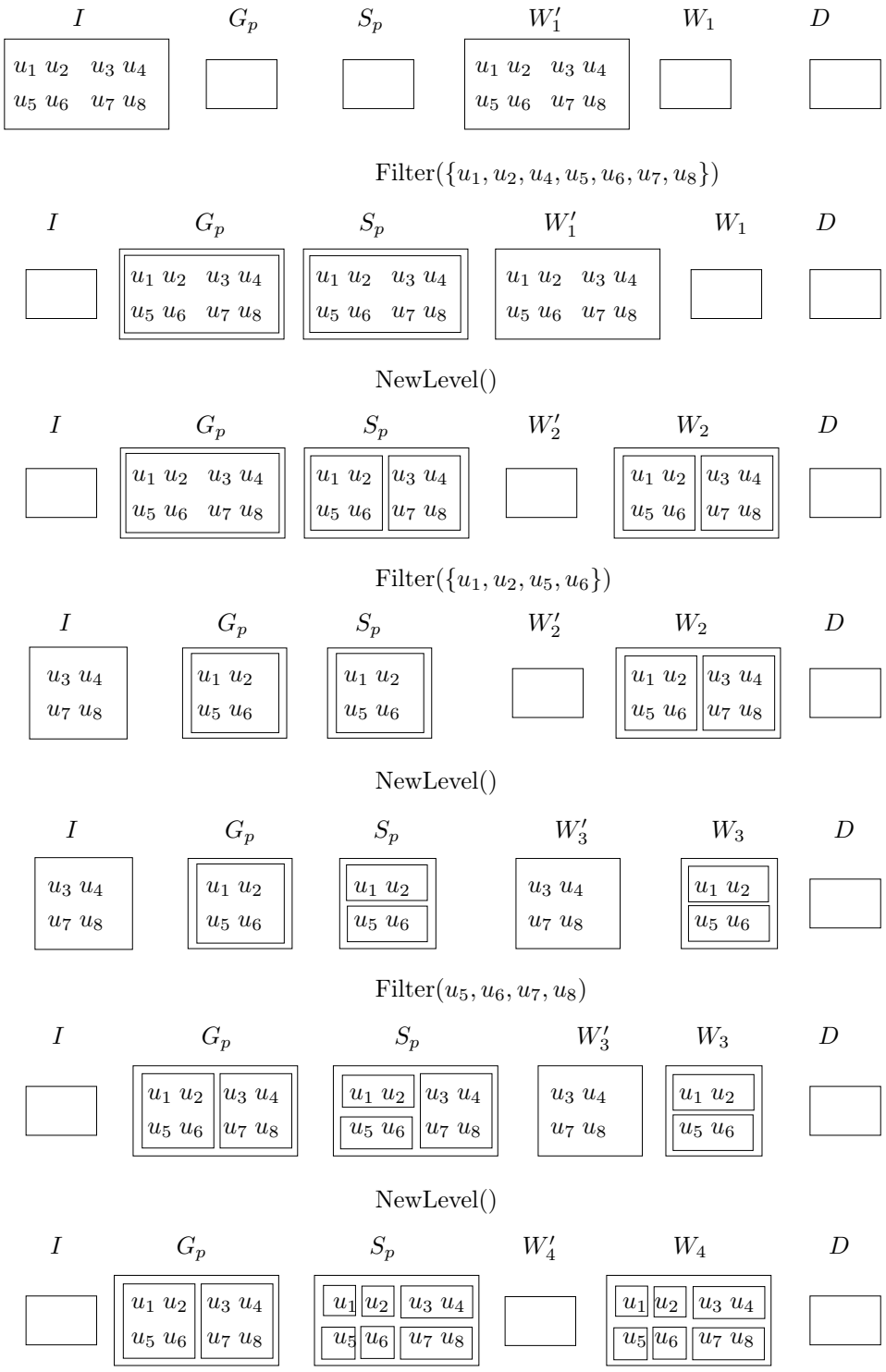


Figure 3.5: An example of traitor tracing using the search tree, illustrating the effect of each algorithm on the states of the user sets, I, G_p, S_p, W_{N_l}, W'_{N_l}. Each set is represented by a block, and blocks within blocks represent sets within sets. The two colluding pirates, u_2 and u_7, are eventually singled out and disconnected, leaving all other users in the innocent set, I.

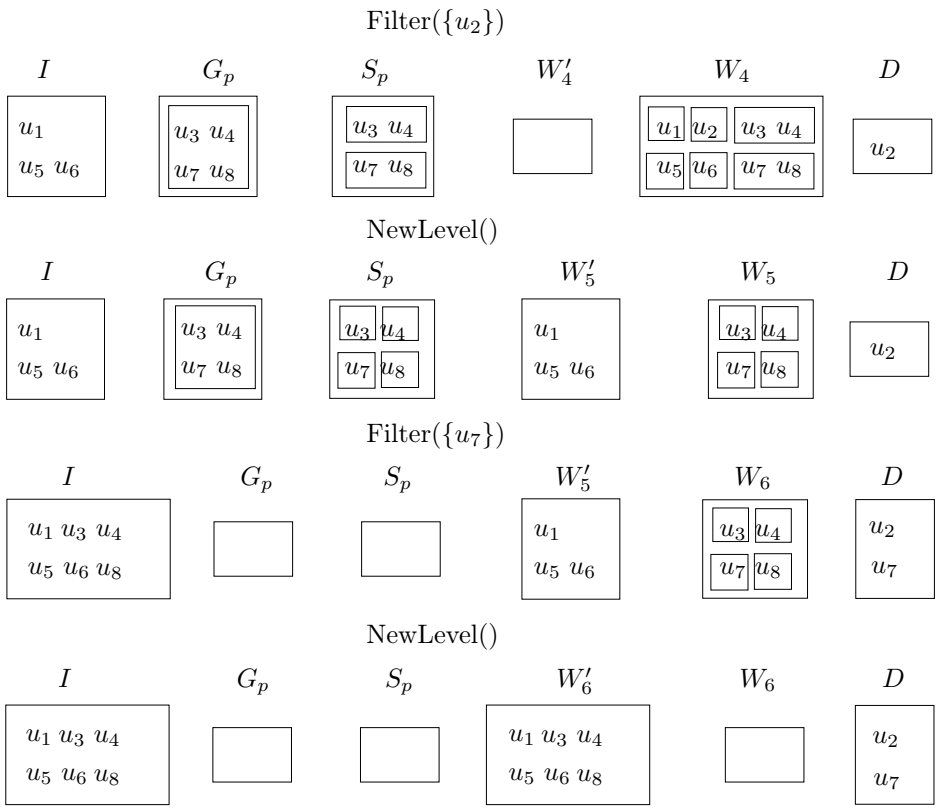


Figure 3.6: Example of traitor tracing using the search tree *(continued from Figure 3.5)*.

3.2.2 Comparison with Fiat and Tassa Search Tree

Our search tree operates very similarly to that of the FT scheme [18], described in Section 2.2.6, except that the tree level generating function in Algorithm 1, and the tree filtering function in Algorithm 3 can be applied to the tree independently of each other, at any time. This means that, whenever marked segments are received from a pirate broadcast, the search tree is updated based on the incriminated set of users. However, while the CP waits to receive delayed pirate segments, the search tree can still continue to narrow the search for pirates.

Immediately before a new video segment is to be broadcast, Algorithm 1 is used to partition U into the subsets that will each receive unique variants. If no set of users has yet been incriminated by a previously received variant, then *every* set that is not assumed innocent is split in half for the next iteration. This is in contrast to the FT scheme, where only a set that has been incriminated by a received variant is split. Our scheme uses the filter function in Algorithm 3 to remove the extraneous user sets from the search tree as soon as it is clear from the received variants that those sets are indeed extraneous. When a user set, P, is incriminated, any extra suspect sets generated by the new level function in Algorithm 1 in an attempt to trace the pirates in P, but which have now been found not to contain P, are returned to I. Although the extraneous user sets are eventually removed from the search tree, the extra sets require more than the minimum number of variants to be generated for each segment. Therefore our scheme uses more than $2N_c + 1$ variants to trace N_c pirates. This trade-off allows the search tree to progress even when feedback from previous segments is delayed, which allows the scheme to disconnect all pirates more quickly than the FT scheme, which wastes time waiting for previous segments.

3.2.3 Segment Marking Using Dynamic Tardos Codes

We now describe how we construct the watermark messages embedded in each variant of each video segment, and how we incriminate a particular set of users when a pirate segment is received. Immediately prior to the broadcast of the k^{th} video segment, our scheme generates a new level of the search tree and generates a unique variant for each non-empty set of users in $W_k \cup \{W'_k\}$. Let c_k be the number of unique variants transmitted during iteration k, given by

$$c_k = \begin{cases} |W_k|, & \text{if } W'_k = \emptyset \\ |W_k| + 1, & \text{if } W'_k \neq \emptyset. \end{cases} \tag{3.2.1}$$

Let $\mathbf{m}^{(W,k)}$ be the unique watermark message assigned to a particular non-empty user set, $W \in W_k \cup \{W'_k\}$, during the k^{th} iteration. Each segment message, $\mathbf{m}^{(W,k)}$, is further partitioned into sub-segment messages, where $\mathbf{m}^{(W,k,i)}$ is the i^{th} sub-segment of $\mathbf{m}^{(W,k)}$. The purpose of this partitioning is to allow

the tracing algorithm to shorten $\mathbf{m}^{(W,k)}$ when it is determined that a shorter code length is sufficient for the k^{th} iteration. The watermark messages for $W \in W_k \cup \{W'_k\}$, and $i \in 1, \ldots, c_k - 1$ are constructed using the dynamic Tardos scheme of Laarhoven et al. [16], which we described in Section 2.2.5 of the previous chapter. We construct the codewords as follows.

1. For each bit position j, $1 \leq j \leq l(c_k, c_k, \epsilon_1, \epsilon_2)$, select ρ'_j independently randomly from a uniform distribution over $[\delta_{\min}, \frac{\pi}{2} - \delta_{\min}]$, where $\delta_{\min} = \delta(c_k, c_k, \epsilon_1, \epsilon_2)]$.

2. Calculate the vector, $\mathbf{p}(c)$, for $c \in \{2, \ldots, c_k\}$, as in step 2 of the construction on page 25.

3. Set $\mathbf{m}_j^{(W,k)} = 1$ with probability $\rho_j = \sin^2 \rho'_j$ for all $1 \leq j \leq l(c_k, c_k, \epsilon_1, \epsilon_2)$, and non-empty user set $W \in W_k \cup \{W'_k\}$.

4. For $c \geq 0$ define

$$l_{\min}(c) = \begin{cases} \min\left\{\left[j' : \sum_{j=1}^{j'} \mathbf{p}(c)_j = l(c, c_k, \epsilon_1, \epsilon_2)\right]\right\} & , \quad c \geq 2 \\ 0 & , \quad c < 2 \end{cases}, \tag{3.2.2}$$

which gives the minimum number of bits after which $\mathbf{m}^{(W,k)}$ can be truncated, where the truncated version is still an ϵ_1-sound, ϵ_2-complete dynamic Tardos codeword for a maximum of c collaborating pirates. For each non-empty user set $W \in W_k \cup \{W'_k\}$, and $i \in 1, \ldots, c_k - 1$, let the i^{th} watermark message sub-segment, $\mathbf{m}^{(W,k,i)}$, be given by

$$\mathbf{m}^{(W,k,i)} = \left[\mathbf{m}_{l_{\min}(i)}^{(W,k)} \quad \mathbf{m}_{l_{\min}(i)+1}^{(W,k)} \quad \ldots \quad \mathbf{m}_{l_{\min}(i+1)}^{(W,k)}\right]. \tag{3.2.3}$$

Figure 3.7 illustrates the partitioning of a watermark segment message, $\mathbf{m}^{(W,k)}$, into sub-segments for $c_k = 5$. When the i^{th} watermarked sub-segment that was originally transmitted in the k^{th} iteration is received, the collusion-attacked watermark message, $\mathbf{m}^{(C,k,i)}$, is decoded from that sub-segment and the following steps are carried out.

1. Append to the other sub-segments from the k^{th} iteration to give the current segment watermark message, $\mathbf{m}^{(C,k)}$, given by

$$\mathbf{m}^{(C,k)} = \left[\mathbf{m}^{(C,k,0)} \quad \mathbf{m}^{(C,k,1)} \quad \ldots \quad \mathbf{m}^{(C,k,i)}\right]. \tag{3.2.4}$$

2. Construct the score vector, $\mathbf{s}'(W)$ for each non-empty user set $W \in W_k \cup \{W'_k\}$ as follows:

$$\mathbf{s}'(W)_j = \begin{cases} +\sqrt{(1-\rho_j)/\rho_j}, & \text{if } \mathbf{m}_j^{(W,k)} = 1 \text{ and } \mathbf{m}_j^{(C,k)} = 1 \\ -\sqrt{\rho_j/(1-\rho_j)}, & \text{if } \mathbf{m}_j^{(W,k)} = 0 \text{ and } \mathbf{m}_j^{(C,k)} = 1 \\ -\sqrt{(1-\rho_j)/\rho_j}, & \text{if } \mathbf{m}_j^{(W,k)} = 1 \text{ and } \mathbf{m}_j^{(C,k)} = 0 \\ +\sqrt{\rho_j/(1-\rho_j)}, & \text{if } \mathbf{m}_j^{(W,k)} = 0 \text{ and } \mathbf{m}_j^{(C,k)} = 0 \end{cases}. \tag{3.2.5}$$

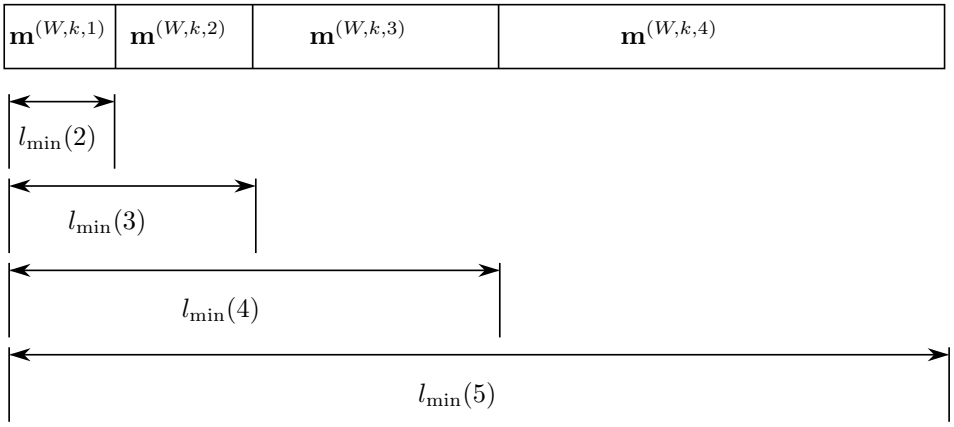


Figure 3.7: Illustration of the partitioning of a segment watermark message, $\mathbf{m}^{(W,k)}$, into sub-segments, where the i^{th} sub-segment is $\mathbf{m}^{(W,k,i)}$. At the beginning of this iteration of the dynamic traitor tracing scheme, five differently watermarked variants are transmitted. The minimum number of watermark message bits required for an ϵ_1-sound, ϵ_2-secure Tardos code for c variants is given by $l_{\min}(c)$. If feedback from previous iterations indicates that, for example, only four variants are being used for the collusion attack, then this iteration will end after $\mathbf{m}^{(W,k,3)}$ is transmitted, since enough watermark message bits will have been transmitted to incriminate at least one of the four variants.

3. For each non-empty user set $W \in W_k \cup \{W'_k\}$, and each value of $c \in \{1, \ldots, c_k\}$, update the accusation score,

$$s_{\text{dyn}}\left(W, c, \mathbf{m}^{(C,k)}\right) = \sum_{j=1}^{l_{\min}(i+1)} \mathbf{s}'(W)_j \mathbf{p}(c)_j, \tag{3.2.6}$$

   and incriminate the user set W if $s_{\text{dyn}}(W, c, \mathbf{m}^{(C,k)}) > \tau(c, c_k, \epsilon_1, \epsilon_2)$ for any c.

If the pirates in C generated the pirate video segment using parts from more than one of the video segments available to them, then the incrimination step of iteration k may incriminate more than one user set as containing one of the pirates. More incriminated user sets increase the number of suspect sets in S_p, which increases c_{k+1}, and means that more differently watermarked variants will need to be generated during the following iteration. Since the number of message bits required by the Tardos codeword construction increases proportionally to c_{k+1}^2, this means that a significantly longer watermark message is needed for iteration $k+1$.

Let P'_k be the set of user sets incriminated by the collusion-attacked message, $\mathbf{m}^{(C,k)}$. When $|P'_k| > 1$, we simply choose one of the incriminated sets. The incriminated set that we choose to use for the search tree filtering function

in Algorithm 3 is given by

$$P = \arg\min_{W \in P'_k}(|W|), \tag{3.2.7}$$

which is simply the incriminated set containing the fewest users. This corresponds to the branch of the search tree that has narrowed down the most on a particular pirate. The advantage of this choice is that it allows singleton user sets to be generated and incriminated quickly. Every time a pirate is incriminated, all other users in the suspect set containing that pirate are moved to the innocent set, I. This reduces the number of suspect user sets, and therefore watermarked variants required for the rest of the algorithm. Fewer watermarked variants require shorter Tardos codes, and the algorithm can trace all pirates using fewer watermark bits.

3.2.4 The Complete Dynamic Traitor Tracing Scheme

Algorithm 4 represents the entire dynamic traitor tracing scheme. The scheme operates on the search tree described in Section 3.2.1. During the k^{th} iteration the sub-segments watermarked with $\mathbf{m}^{(W,k,i)}$ are transmitted in sequence to each set $W \neq \emptyset \in W_k \cup \{W'_k\}$ of users. As the transmission of segment k progresses, previous watermarked segments may be received (after some delay after their transmission). The received segments are used by the filter function in Algorithm 3 to narrow down the sets of suspect users in S_p, thereby adding users to the innocent set, I. This may result in a reduction in $|S_p|$, which is the estimated maximum number of differently watermarked variants available to the pirates in C. The transmission of segment k ends when the number of watermark message bits transmitted during the segment is greater than or equal to the number of bits required by the dynamic Tardos code construction for ϵ_1-sound, ϵ_2-complete incrimination when at most $|S_p|$ variants are used for collusion. This prompts transmission of the next watermarked segment to begin. Figure 3.8 illustrates the process of tracing one pirate within a set of four users.

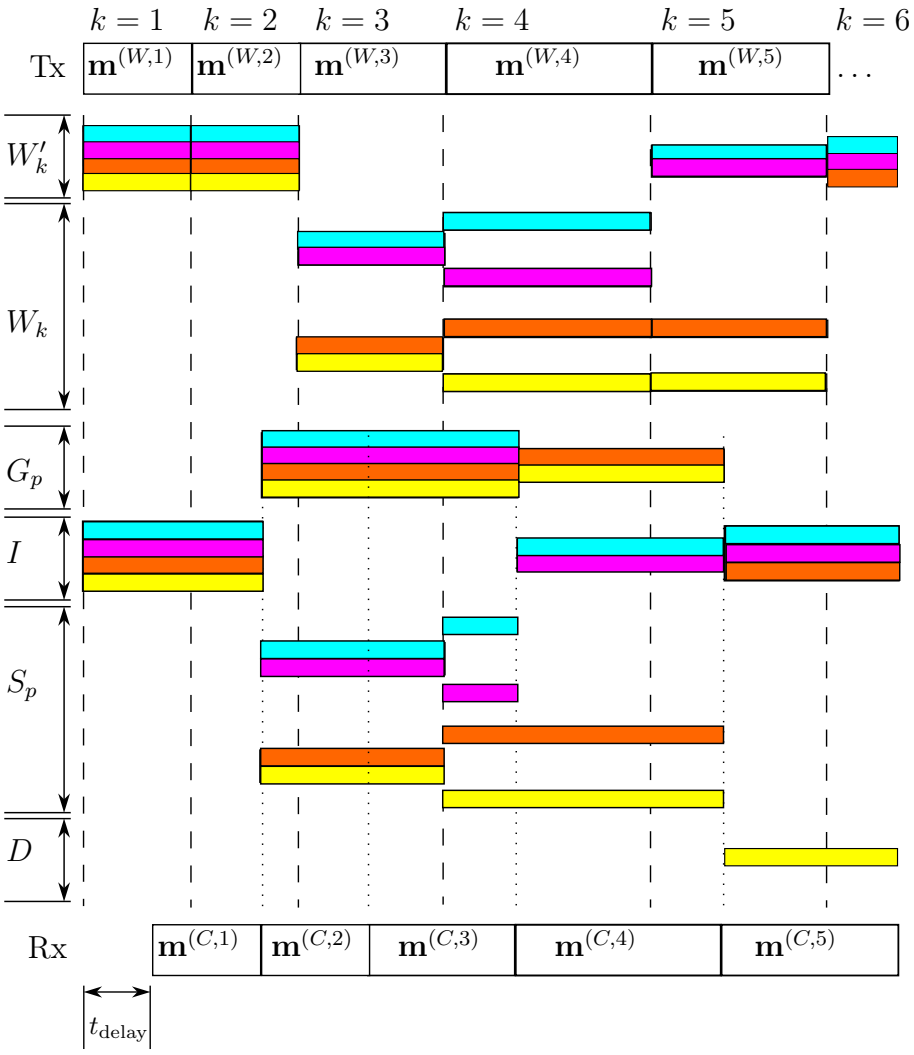


Figure 3.8: Diagram illustrating the operation of the delay tolerant search tree when tracing a single pirate among four users. Each coloured bar represents a user. The bar labelled "Tx" represents the transmitted video segments containing the watermark messages, $\mathbf{m}^{(W,k)}$. The dashed lines show the start of the k^{th} segment. The bar labelled "Rx" represents the pirate video segments, received by the CP after a delay of t_{delay}, containing the watermark messages, $\mathbf{m}^{(C,k)}$. At the beginning of each iteration, each suspect set of users is split in half until a singleton set is generated. If the user in the singleton set is incriminated then that user is immediately disconnected.

Algorithm 4 The Complete Dynamic Traitor Tracing Scheme.

for all $k \in \{1, 2, 3, \ldots\}$ **do**
  NewLevel
  Set $i = 0$.
  Generate $\mathbf{m}^{(W,k)}$ for all $W \in W_{N_l} \cup \{W'_{N_l}\}$, $W \neq \emptyset$.
  while $i <= |S|$ **do**
    for all $W \neq \emptyset \in W_{N_l} \cup \{W'_{N_l}\}$ **do**
      Watermark the host segment, $v^{(k,i)}$, with $\mathbf{m}^{(W,k,i)}$ to give $v^{(W,k,i)}$.
      Transmit $v^{(W,k,i)}$ to all users in W.
    end for
    for all received pirate segments, $v^{(C,k')}, k' \leq k$ **do**
      Extract watermark message, $\mathbf{m}^{(C,k)}$.
      Incriminate user set $P = \arg\min_{W \in P'_{k'}}(|W|)$.
      Filter(P)
    end for
    Increment i.
  end while
end for

Chapter 4

Detail Design

We now describe the details of the implementation of the designs presented in the previous chapter. Since the intention is to design a prototype scheme, we implemented the algorithms in software, using the Python programming language [49]. While Python is generally slower at high-computation tasks than C or Fortran, it allows for faster and simpler prototyping [50]. Additionally, the NumPy mathematical Python library [51] provided the tools necessary to quickly write efficient mathematical routines.

4.1 The Digital Watermarking Scheme

The prototype digital watermarking scheme, described in Section 3.1 of the previous chapter, requires a number of parameters, which can affect its performance in terms of security, fidelity, and robustness. Here we discuss the values we chose for these parameters. In some cases the parameters are chosen for practicality or simplicity, and therefore do not necessarily result in optimal performance. However, the objective of this thesis is to evaluate a working prototype, and the optimisation of parameters is left as future work.

4.1.1 Watermarking Parameters

Section 3.1 described a method of watermarking a single colour component of a digital video. For the implementation we chose to watermark digital videos where pixel colour values are represented in the $Y'C_bC_r$ colour space. We chose $Y'C_bC_r$ for two reasons. Firstly, it is the colour space used in MPEG encoded videos, as well as uncompressed video formats like YUV and Y4M. This removes the need for colour space conversion, which can add computational complexity to the encoding and decoding operations. Secondly, $Y'C_bC_r$ separates brightness and colour information. This is advantageous since the human eye is generally more sensitive to brightness information than to colour information [7]. In order to reduce the perceptibility of the watermark signal

we applied the watermarking scheme to the two colour channels (C_b and C_r), and left the luma channel unmodified.

We chose to implement the watermarking scheme using 1080p HD videos with a luma channel spatial resolution of $X_l = 1920$ and $Y_l = 1080$ pixels, since this is a standard format for high definition video, and is the highest resolution video that is currently widely available [52]. We chose to use a chroma subsampling ratio of 4:2:0, because it is commonly used in MPEG video [35]. This resulted in a colour channel spatial resolution of $X = 960$, and $Y = 540$ pixels. We chose the encoding and decoding frame rate, r_{dec}, to be 29.97 FPS, which is the NTSC standard video frame rate [35]. We chose the length of a watermark block, T_b, to be 60 frames, which is approximately equal to 2 seconds of video playback.

We chose the resolution of the STDM chip signals to be $X_s = 60$, $Y_s = 33$, and $T_s = 15$. We let T_s be an integer factor of T_b to simplify the calculation of the downsampled signal values used by the encoder and decoder. We chose to use a watermark signal bit rate of $N_b = 64$ bits per video block for each colour channel, which corresponds to approximately 64 bits of watermark signal per second of video. Our choice of chip signal resolution, video block size and watermark bits per video block result in $N_s = 464$ coordinates where the chip signal corresponding to each bit is non-zero. According to (3.1.15), this allows us to use a quantiser step size of $\Delta = 928$ for STDM encoding and decoding.

4.1.2 Decoding Parameters

The implementation of the watermark decoder requires parameters that affect the functionality of the synchronisation process. Section 3.1.6 on page 35 described the temporal synchronisation process, where the decoder estimates the effective frame rate scaling factor, r_k, by sweeping over values from the set R. In this implementation we chose

$$R = \{1.05, 1, 0.95, 0.9, 0.85, 0.8\}. \tag{4.1.1}$$

Section 3.1.7 on page 37 discussed the estimation of the gain factor applied to the watermarked video signal. In order to simplify the process of countering gain attacks we restrict the value of the gain attack factor, $\alpha(t)$ to $0.85 \leq \alpha(t) \leq 1.15$. We assume that values of $\alpha(t)$ outside these bounds will produce unacceptable degradation to the video and need not be considered for a practical video watermarking scheme. These bounds simplify analysis by limiting the range of possible values that the decoder must consider. The decoder estimates the gain factor by generating a histogram of the dither modulated watermark values. In our implementation the decoder uses a value of $f_\Delta = 50$ in (3.1.27) to determine the histogram bin size.

4.1.3 Video Processing Tools and Formats

For frame rate conversion and spatial resizing of videos, we used avconv, an open-source command line video processing tool built using the multimedia library, *libav* [53]. We also used this tool to perform the frame rate scaling required by the temporal synchronisation process. In our implementation we combined this step with the initial conversion of the received video to a frame rate of r_{dec}. Upon receiving a video $p^{(u)}[x, y, t]$ the decoder simply used avconv to generate, for each $\hat{r}'_k \in R$, the version of the received video, $p^{(u)}[x, y, t/\hat{r}'_k]$ with a frame rate of $r_{\text{dec}}/\hat{r}'_k$.

Our watermarking scheme is designed to watermark uncompressed digital videos. We used the YUV4MPEG2 (Y4M) video format to store the uncompressed videos, because of its simple file structure, which can easily be read and written by Python scripts, and because it can easily be compressed into other formats like MPEG. A Y4M video file consists of a header that includes information about the video (including its width and height in pixels, and its frame rate), followed by the video frames. Each frame is stored as an $X_l \times Y_l$ array of luma values, followed by two $X \times Y$ arrays of colour values. Each element of these arrays is represented by a byte value.

4.2 The Dynamic Traitor Tracing Scheme

We implemented the traitor tracing scheme by writing a Python script to carry out the algorithms described in Section 3.2 of the previous chapter. Since our goal in this thesis is to prove the concept of the scheme, we designed the implementation to work as part of a simulation of a broadcast environment where a group of colluding pirates redistributes a pirate copy produced by a collusion attack. We therefore simulated the users as well as the actions of the colluding pirates.

The dynamic Tardos traitor tracing codes, which we used to watermark the segment variants of our dynamic traitor tracing scheme, were constructed using parameters that minimised the lengths of the codewords encoded into each segment. In Section 2.2.5 of Chapter 2 we described the constraints involved in optimising the codeword lengths. We carried out the optimisation procedure numerically using the optimisation routines available in the SciPy scientific library for Python [54]. For our implementation we generated ϵ_1-secure codes where $\epsilon_1 = 0.01$.

Chapter 5

Testing

In this chapter we test the robustness of our watermarking scheme against some attacks to which a watermarked video might be subjected prior to re-distribution as a pirate copy. Section 5.1 focusses on testing the robustness of the digital watermarking scheme, described in Section 3.1. Robustness against collusion attacks is provided by the dynamic traitor tracing scheme, described in Section 3.2, which we test in Section 5.2.

5.1 The Watermarking Scheme

In this section we test the performance of the digital watermarking scheme. This includes robustness against attacks that can be applied to a watermarked video when only one video is available to the attacker. These attacks include addition of noise, lossy video compression, resizing, frame rate changes, frame deletion, amplitude scaling, recompression, and cropping.

5.1.1 Methodology

In order to test the robustness of the watermarking scheme against non-collusion attacks we used the scheme to digitally watermark five different uncompressed digital video clips, each consisting of 300 video frames. The host video clips were stored in Y4M format with a (luma channel) spatial resolution of $H_l = 1080$ pixels, $W_l = 1920$ pixels, a frame rate of 29.97 FPS, and a chroma subsampling ratio of 4:2:0. We generated ten different watermarked versions of each host video for a total of fifty test videos. In each version, a different randomly generated watermark message was encoded, using a different randomly generated chip function. The watermarked test videos were generated using the parameters described in Section 4.1.1 on page 50.

We processed the watermarked videos using avconv to carry out the attacks for each test, except for the AWGN, gain, and random frame deletion attacks, where we used software we wrote in Python to carry out the attacks. Each

attacked video was then input to the decoder, which generated a decoded watermark message for that video. Since the prototype decoder is designed to read uncompressed Y4M video files, we used avconv to decompress any compressed attacked videos into Y4M format before decoding. The bit error rate (BER) between each encoded message and its corresponding decoded message was then calculated. For an arbitrary watermark message, $\mathbf{m}$, encoded in a watermarked video, and $\hat{\mathbf{m}}$, the decoded watermark message, the BER is given by

$$\text{BER} = \frac{\text{number of incorrect bits in } \hat{\mathbf{m}}}{\text{total number of bits in } \mathbf{m}}. \tag{5.1.1}$$

When testing robustness we consider the two requirements for correct decoding of watermark message bits described in Section 1.2.2 on page 3. The first requirement is that the part of the watermark signal corresponding to each message bit is not corrupted. The second requirement is that the decoder is able to achieve both spatial and temporal synchronisation with the encoder, so that each bit is decoded from the correct part of the watermarked signal. For our STDM-based scheme, the decoder also has to estimate the possible pixel value scaling factor applied to the watermarked video. We include this gain attack estimation step when we refer to synchronisation.

In order to test the effectiveness of the synchronisation algorithm of our decoder, for each attack tested we generated two different decoded messages, and determined the BER for each of the two. The first message is generated when our decoder is allowed to automatically synchronise with the encoder, using the methods described in Sections 3.1.6 and 3.1.7. The second message is generated by providing the correct synchronisation parameters to the decoder. We say that the first message is generated using *decoder synchronisation*, and the second message is generated using *manual synchronisation*. The BER under manual synchronisation indicates the rate at which watermark message bits are actually corrupted by an attack. If the BER under decoder synchronisation is higher than under manual synchronisation, then the decoder was unable to synchronise properly on parts of the watermark signal.

5.1.2 Document-to-Watermark Ratio

Section 1.2.2 on page 3 discussed document-to-watermark ratio (DWR) as a means of determining the effect of the watermark signal on the fidelity of the host signal. The objective of this test is to measure the average document-to-watermark ratio of the watermarking scheme, using the parameters discussed in Section 4.1.1 on page 50.

For each watermarked test video, we calculated the DWR according to the definition in (1.2.1) on page 4. We measured the video signal amplitudes in pixel values, which are integers in the range $\{0, \ldots, 255\}$. Table 5.1 shows the average variance of the host video signals, the average variance of the watermark signals, and the average DWR of the watermarked host videos for

Colour channel	Host variance	Watermark variance	DWR
C_b	194.81	0.487	400.44
C_r	67.50	0.485	139.31

Table 5.1: Host signal variance, watermark signal variance, and document-to-watermark ratio of the test videos. The variances are given in units of squared pixel value, where the range of a pixel value is $\{0, \ldots, 255\}$.

the C_b and C_r colour channels. The average variance of the watermark signal is 0.487 for the C_b channel, and 0.485 for the C_r channel. The host signal variance of the C_b channel is 197.81. This is more than double the host variance of the C_r channel, which is 67.5. The DWR for the C_b channel is therefore higher than that of the C_r channel. This result may explain the difference between the robustness of the C_b channel and C_r channel watermarks observed in some of the tests that follow.

5.1.3 Basic System Test

The objective of the first test of the watermarking scheme is to show that the watermarking scheme works as expected. We do this by showing that the watermark extractor is capable of extracting the watermark message from an uncompressed digital video that has been watermarked by the encoder.

For this test, we input the watermarked test videos, described in Section 5.1.1, directly into the watermark decoder, without subjecting the videos to any attack. We calculated the BER of the decoded watermark messages according to the procedure in Section 5.1.1.

Since the watermarking scheme uses side information at the transmitter to prevent interference between the host signal and the watermark signal, and since the watermarked video is not modified after watermark encoding, the channel noise is expected to be zero. Therefore, the expected result of this experiment is that the decoder is able to decode the watermark messages from the unmodified, watermarked videos with a BER of zero.

The result of this test was that the BER was exactly zero under both manual and decoder synchronisation for every watermarked test video. This result agrees with what was expected. In addition to calculating the BER, we generated two graphs that demonstrate the action of the decoder synchronisation. Figure 5.1 shows the decoder-calculated likelihood, $\beta_k(\hat{r}'_k, \hat{\theta}'_k)$, that the k^{th} watermarked video block lies at frame offset $\hat{\theta}'_k$ with a frame rate scaling factor of $\hat{r}'_k = 1$. The distinct peaks at $\hat{\theta}'_k \in \{0, 60, 120, 180, 240\}$ show that the decoder detected the watermarked video blocks using the correct decoding parameters. Figure 5.2 shows the same graph, but for $\hat{r}'_k = 0.8$. The likelihood scores are low since the frame rate scaling factor is incorrect. The results confirm that the watermarking scheme works correctly.

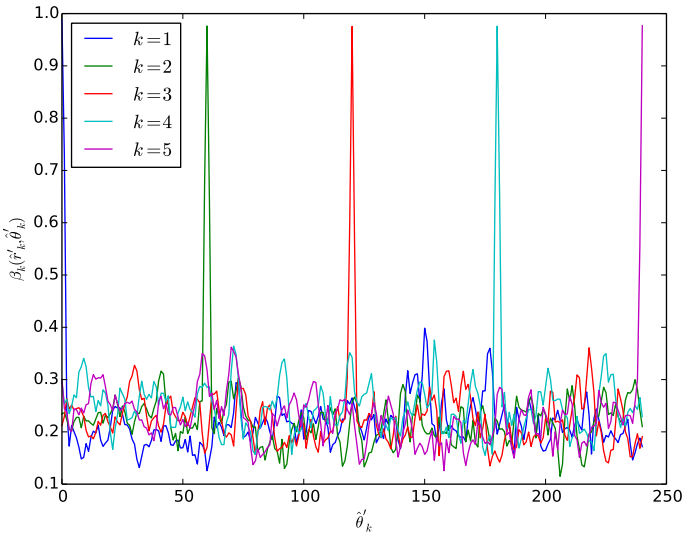


Figure 5.1: $\beta_k(\hat{r}'_k, \hat{\theta}'_k)$ vs. $\hat{\theta}'_k$ for the C_r channel of one of the test videos, where $\hat{r}'_k = 1$. The distinct peaks indicate a high certainty of these synchronisation parameters, which is expected since these are the correct parameters.

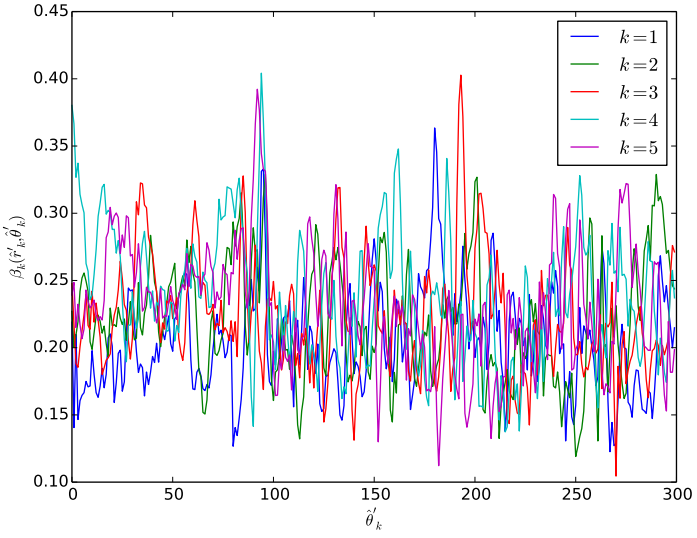


Figure 5.2: $\beta_k(\hat{r}'_k, \hat{\theta}'_k)$ vs. $\hat{\theta}'_k$ for the C_r channel of one of the test videos, where $\hat{r}'_k = 0.8$. The lack of distinct peaks indicates a low certainty of these synchronisation parameters, which is expected since $\hat{r}'_k = 0.8$ is a poor estimate.

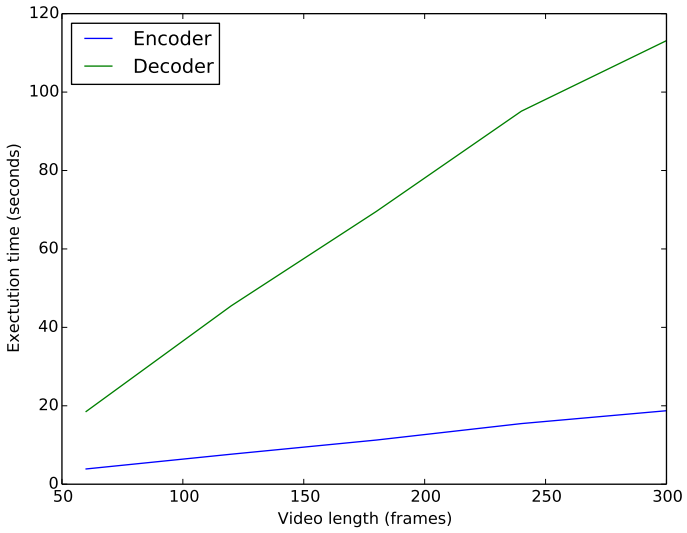


Figure 5.3: Execution time of the encoder and decoder vs. video length.

5.1.4 Computational Complexity

The objective of this test is to determine the time taken to encode and decode a watermark, as well as the memory required. This gives us an indication of the type of hardware required to implement the encoder and decoder.

For this test, we watermarked ten versions of each of the five host videos, while recording the computation time and memory used by the watermark encoder process. The watermarked videos were passed directly to the watermark decoder, without subjecting the videos to any attack. The computation time and memory usage of the decoding process were also measured. The encoder and decoder algorithms, written in Python, were run on a desktop personal computer equipped with a 3.4 GHz quad-core processor and 32 GB of memory.

Figure 5.3 shows the execution time of the encoder and decoder software as the length of the host video increases. The execution time of the encoder increases linearly with video length, at an average encoding rate of 16.2 FPS. This encoding rate is half of the 29.97 FPS necessary for real-time watermark encoding. Optimisation of the encoding algorithm, and implementation of the encoder in a compiled language like C, may allow the encoder to watermark videos in real-time. This is useful for watermarking of live video streams, where content is not available prior to broadcast.

The execution time of the decoder also increases roughly linearly with video length, but at a much lower average rate of 2.8 FPS. One reason for slow execution is that the sweeping of parameters during decoder synchronisation was executed sequentially. In a broadcast setting, using dynamic traitor tracing, watermark decoding also needs to be performed in real-time. The slow performance of the decoder may be acceptable if watermark decoding is carried

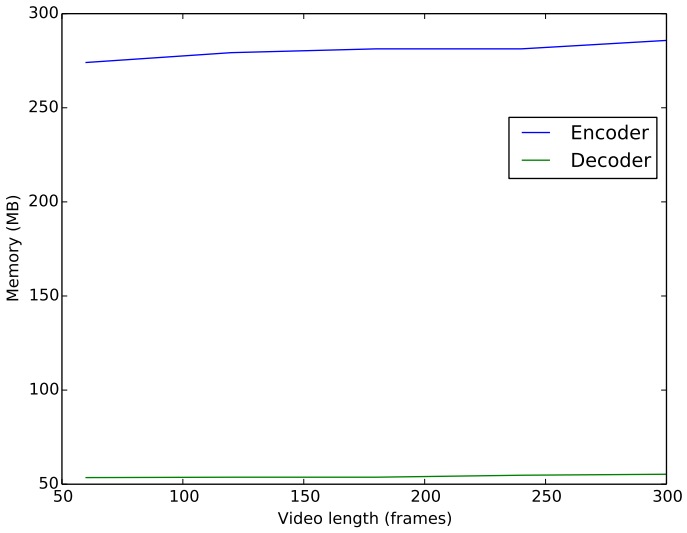


Figure 5.4: Memory usage of the encoder and decoder vs. video length.

out by the CP on powerful hardware that runs a parallelised version of the decoder software.

Figure 5.4 shows the memory usage of the encoder and decoder software as the length of the host video increases. The decoder memory usage increases roughly linearly from 54 MB for a 60-frame video to 56 MB for a 300-frame video. The encoder memory usage increases roughly linearly from 274 MB for a 60-frame video to 286 MB for a 300-frame video. The higher memory usage of the encoder is due to the encoder storing an entire block of the host signal in memory while the watermark signal is generated and added to the host. The decoder simply reads one video frame at a time from the watermarked video.

5.1.5 AWGN Attack

A pirate may add noise to a watermarked video in an attempt to inhibit the decoder's ability to decode the watermark signal correctly. The objective of this test is to evaluate the robustness of the digital watermarking technique against the addition of zero-mean Gaussian noise to the watermarked video. We refer to this attack as the additive white Gaussian noise (AWGN) attack.

For this test, we added zero-mean Gaussian noise to the pixel values of both the C_b and C_r colour channels of each of the watermarked test videos. We then input the AWGN-attacked videos to the watermark decoder, and calculated the BER of the manually synchronised, and decoder synchronised messages. We generated attacked videos with noise standard deviations of 8, 16, 24, and 32. These standard deviations are measured in pixel values, which lie in the range $\{0, \ldots, 255\}$.

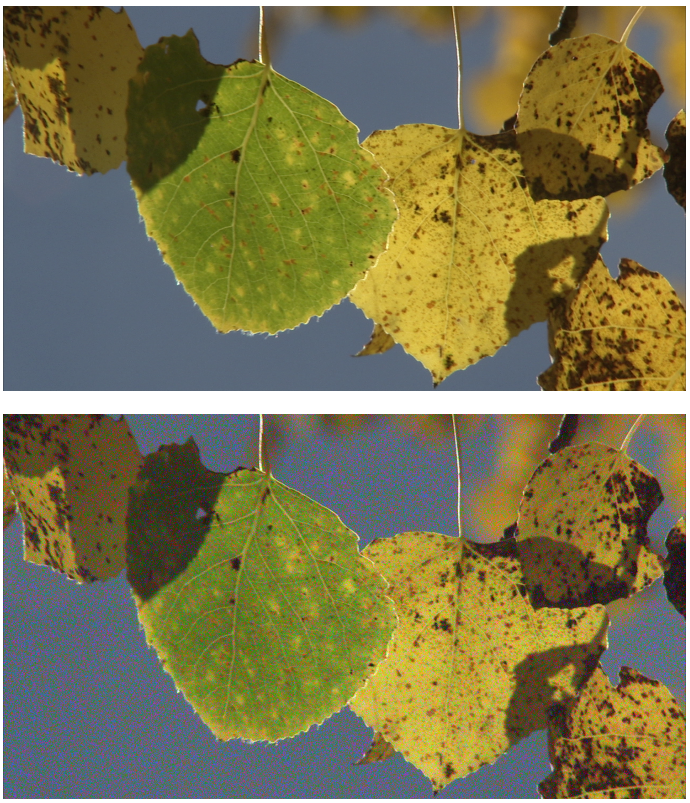


Figure 5.5: A watermarked video frame (top), and the AWGN-attacked version of that frame with standard deviation $\sigma = 32$ (bottom).

Section 2.1.6 on page 14 discussed the robustness of the STDM technique against the addition of a noise signal. We therefore expected that the watermarking scheme would be very robust against the AWGN attack, and that the BER would be very low. The results of the test showed that at each of the noise standard deviations tested, the average BER for both colour channels is zero, under both manual and decoder synchronisation, indicating that the scheme is indeed very robust against the AWGN attack. Figure 5.5 shows an AWGN-attacked video frame with the corresponding original watermarked frame for comparison, in order to illustrate the level of noise to which the scheme is robust. Additionally, in Section 5.1.2 the average variance of the test video signals was found to be 13.96 for the C_b channel and 8.22 for the C_r channel. The highest noise standard deviation tested was therefore more than double the average standard deviation of the host signals, while the BER was zero.

An undesirable side effect of the AWGN attack for the attacker is that it reduces the ability of video compression algorithms to compress the attacked video to a reasonable size, while maintaining reasonable quality. This is because compression algorithms reduce the size of video files by exploiting redundancy between video frames. Adding Gaussian noise to a video adds

variation between pixels that reduces the amount of redundancy in the video, and therefore reduces the effectiveness of video compression. It would be impractical for a pirate to attack the watermarking scheme using AWGN since the resulting pirate copy would be of poor quality and poorly compressed.

5.1.6 Lossy Compression

The watermarking scheme is designed to process uncompressed digital video, but an uncompressed format is impractical for storage and distribution of digital video. This is because the pixel-by-pixel representation of the video in uncompressed format requires a playback bit rate of about 700 megabits per second (Mbps). A thirty-minute video at this bit rate would require 167 gigabytes (GB) of storage. Video compression algorithms are used to reduce the number of bits used to represent a digital video signal. Lossy compression algorithms trade off a loss in video quality for reduced bit rate.

In a practical media distribution scheme the CP would store and distribute compressed digital videos. Therefore, in order for our prototype watermarking scheme to be feasible for such a scheme, it should be robust against lossy video compression. The objective of this test is to evaluate the robustness of the watermarking scheme when the uncompressed watermarked video output by the encoder is compressed using a lossy video compression algorithm. Since the decoder is designed for uncompressed video, any received video is decompressed prior to decoding.

We used avconv to compress the watermarked videos into MP4 format using the h.264 [55] encoding algorithm. We tested the scheme at fixed video bit rates of 30, 25, 20, 10, 5, 2, and 1 Mbps. Figure 5.6 shows an example of the visual effect of compression on the watermarked video for a some of these bit rates.

For this test, we expected that the BER would be low when the compressed bit rate is high, and that, as the bit rate decreases, the BER would increase. We also expected that the degradation of the STDM watermark signal due to compression would cause the decoder to lose synchronisation at compressed bit rates where the watermark message can still be decoded correctly under manual synchronisation. This was expected to result in a greater BER under decoder synchronisation than under manual synchronisation for a particular compressed bit rate.

Figure 5.7 shows the decoder BER of the C_b and C_r colour channels, for both manual synchronisation and decoder synchronisation. Under manual synchronisation, the BER of the C_b channel is zero for compressed bit rates of 10 Mbps and above, and the BER of the C_r channel is zero for bit rates of 15 Mbps and above. Under decoder synchronisation errors start to occur at 15 Mbps for the C_b channel and at 25 Mbps for the C_r channel. As expected, the BER is greater under decoder synchronisation than under manual synchronisation for each bit rate. The BER of the C_r channel is also consistently worse

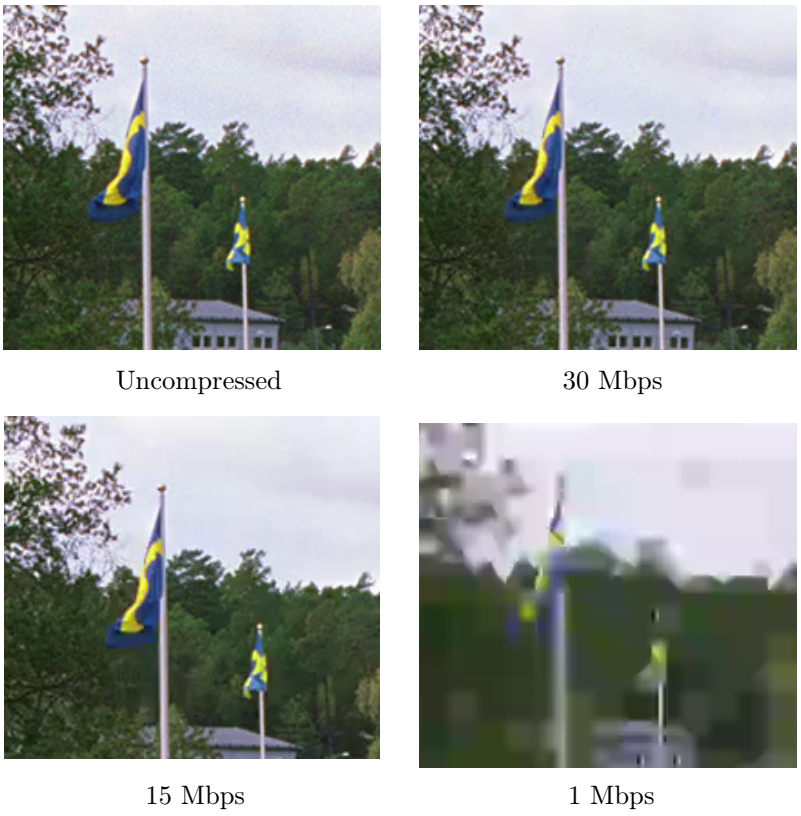


Figure 5.6: Example of the visual effect of different compressed bit rates on the watermarked video. At 15 Mbps, some small patches of blurring are noticeable. At 1 Mbps, the frame contains only severely blurred blocks of colour.

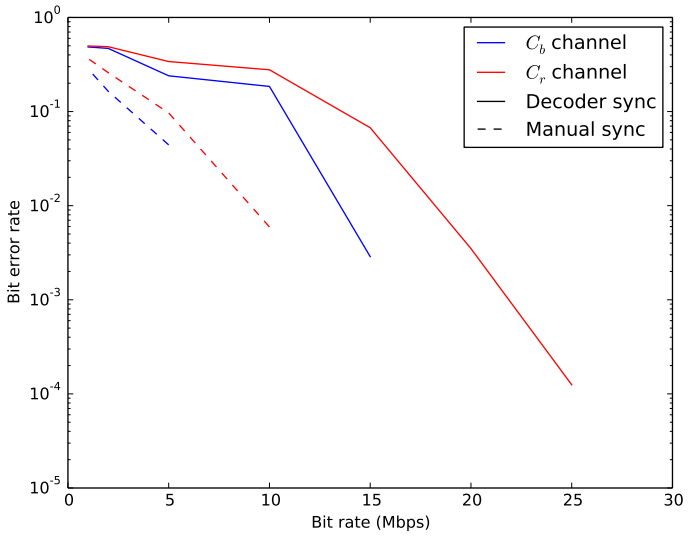


Figure 5.7: Average bit error rate of the C_b and C_r colour channels vs. compressed bit rate.

than that of the C_b channel for both decoder and manual synchronisation.

For the rest of the tests in this section, we carried out the attacks on both the uncompressed watermarked test videos, as well as a compressed version of each watermarked test video. The tests on uncompressed video are intended to show that the system works under ideal conditions, while the tests on the compressed versions of the videos are intended to give an indication of the performance of the scheme if it were implemented in a real-world digital video distribution system. We chose the 30 Mbps, h.264-encoded MP4 format since the average BER of the watermark signal in both colour channels is zero when this compression is used. For the attacks on the uncompressed videos, the attacked videos were also left uncompressed. For the attacks on the compressed videos, the attacked videos were re-compressed to the same format as the watermarked videos at 30 Mbps.

5.1.7 Spatial Resizing

A watermarked video may be resized in order to make it more suitable to a particular display device. A pirate may also resize a watermarked video in an attempt to desynchronise the watermark decoder or to destroy watermark information that is encoded in higher spatial resolutions. The objective of this test is to determine the robustness of the watermarking scheme against resizing of the watermarked video. For this test, we assumed that the spatial resolution of the watermarked video is the highest resolution that is useful for display, and we only considered resizing operations that reduce the spatial resolution of the watermarked video.

We generated five resized versions of each of the compressed and uncompressed watermarked test videos. We resized the videos to spatial resolutions of 1366×768, 1280×720, 960×540, 720×480, and 320×240. We then passed each resized video to the watermark decoder and calculated the BER of the decoded messages under manual and decoder synchronisation.

Since the watermark signal has a much lower spatial resolution than the video signal, we expected that resizing would still preserve the watermark signal well enough to be decoded correctly. We therefore expected that the BER of the resized videos would be low. The results of the test on the uncompressed watermarked videos showed a BER of zero under both manual and decoder synchronisation at all of the resolutions tested. This indicates that the low resolution watermark signal as well as the decoder synchronisation algorithm are robust against resizing when the video is uncompressed.

Figure 5.8 shows the BER vs. spatial resolution of the C_b colour channel when the compressed watermarked videos were tested. Under manual synchronisation, the BER is below one percent for resolutions above 720×480. At the lowest resolution tested, 320×240, the watermark signal is severely corrupted, with a BER of 0.35. The decoder synchronisation performed poorly, with a BER of 0.39 at the highest resolution tested, 1366×768. Figure 5.9 shows

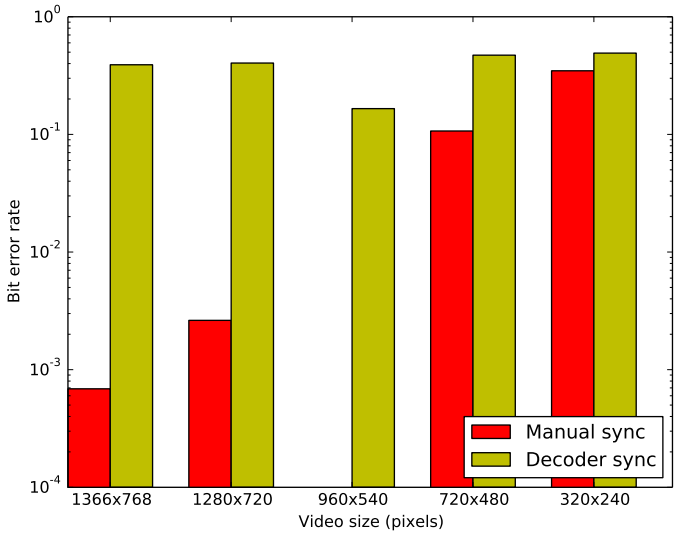


Figure 5.8: Average bit error rate vs. spatial resolution of the C_b channels of the compressed watermarked videos.

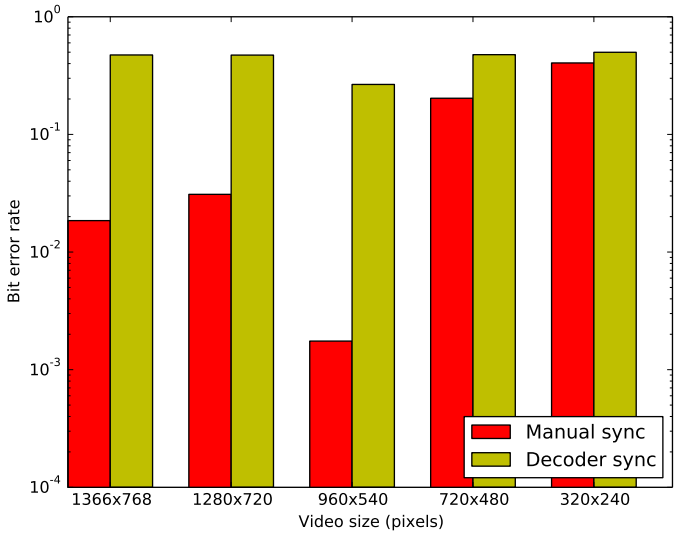


Figure 5.9: Average bit error rate vs. spatial resolution of the C_r channels of the compressed watermarked videos.

the result for the C_r colour channel. The BER under manual synchronisation is higher than that of the C_b channel, but still below four percent for resolutions above 720×480. At 320×240, the BER is 0.41. Again, the decoder synchronisation performed poorly, with a BER of 0.47 at 1366×768.

For both colour channels of the compressed watermarked videos, the BER under decoder synchronisation is much higher than under manual synchronisation, and the lowest bit error rates occur at exactly half the spatial resolution of the original watermarked video, 960×540. This is likely due to the fact that when resizing a compressed video, noise is added to the video signal by the compression algorithm when the resized video is resampled. The loss due to resampling is less when resizing by an integral factor [35].

5.1.8 Change of Frame Rate

The frame rate of a watermarked video may also be altered to suit a particular display device. For example, in the NTSC standard used in North America, 29.97 FPS is used, whereas in the PAL standard used in much of Europe and Africa, 25 FPS is used [35]. It is useful for the watermarking scheme to be robust against changes of frame rate, so that the frame rate of existing watermarked videos can simply be converted for distribution using a particular standard, instead of having to generate different watermarked copies for each standard. The objective of this test is to determine the robustness of the watermarking scheme when the frame rate of the watermarked video is changed.

We generated four versions of each uncompressed watermarked test video with frame rates of 10, 15, 20, and 25 FPS, as well as four versions of each compressed watermarked test video with the same frame rates. We input each of these attacked videos to the decoder and calculated the BER for the manually synchronised and decoder synchronised decoded messages. For this test, we expected that the BER would increase as the frame rate of the videos decreases. We also expected that the BER would be very low at 25 FPS since this is near to the original test video frame rate of 29.97 FPS.

For the uncompressed versions of the test videos, we found the average BER to be zero at each frame rate tested under both manual and decoder synchronisation. This indicates that the temporal resolution of the watermark signal is low enough that the interpolation performed by the video encoder when changing the frame rate of the uncompressed watermarked videos has little effect on the STDM signal.

Figure 5.10 shows the BER vs. frame rate of the C_b channel of the compressed watermarked test videos. Under manual synchronisation, the BER is 1.25×10^{-4} at 25 FPS, and increases with decreasing frame rate to 0.20 at 10 FPS. The BER under decoder synchronisation is 0.1 at 25 FPS, and is higher than the BER under manual synchronisation for each frame rate except 15 FPS. Figure 5.11 shows the decoder BER vs. frame rate of the C_r channel of the compressed watermarked test videos. The BER under manual synchroni-

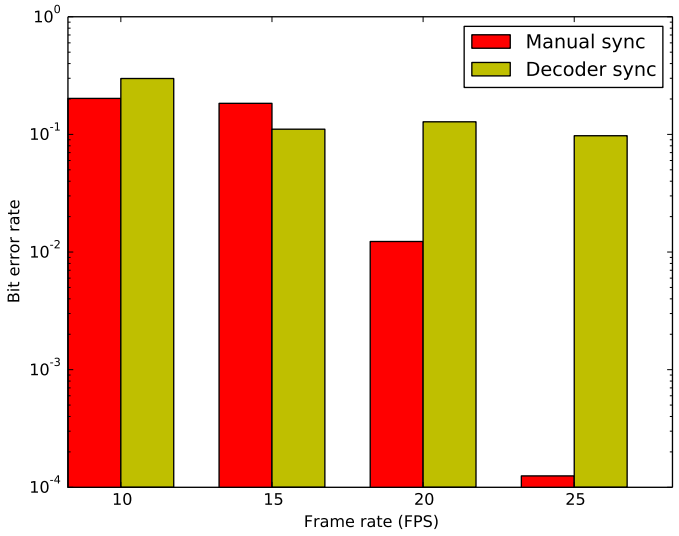


Figure 5.10: Average bit error rate vs. frame rate of the C_b channels of the compressed watermarked videos.

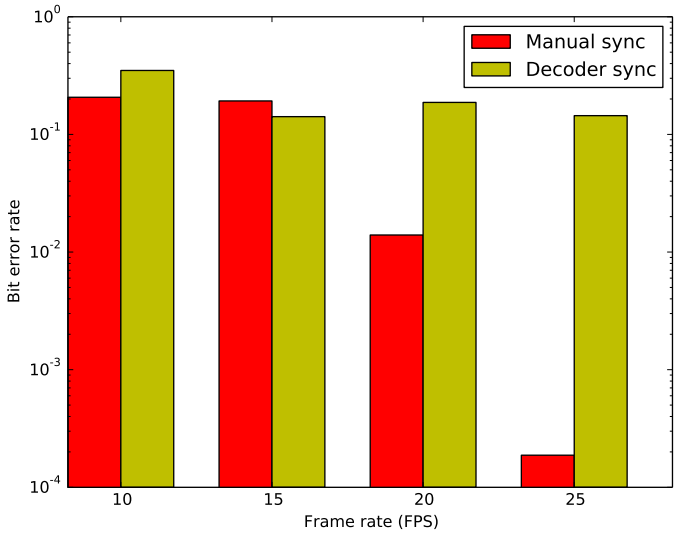


Figure 5.11: Average bit error rate vs. frame rate of the C_r channels of the compressed watermarked videos.

sation is low at 25 FPS, with a value of 1.88×10^{-4}, and increases to 0.20 at 10 FPS. The BER under decoder synchronisation is similar to that of the C_b channel, with a value of 0.14 at 25 FPS.

At 15 FPS, the decoder BER is lower under decoder synchronisation than under manual synchronisation for both colour channels. This is likely due to the locations and sizes of the watermarked blocks within the video being slightly altered by the resampling of the video compression algorithm when the frame rate is changed. In this case, the decoder synchronisation was able to find a better estimate of the actual parameters of the watermark signal than the parameters with which the watermark was embedded.

5.1.9 Random Frame Deletion

The objective of this test is to evaluate the robustness of the watermarking scheme against the random deletion of video frames from a watermarked video. This type of attack may be carried out by a pirate in an attempt to desynchronise the watermark decoder or to destroy any watermark message bits encoded in the deleted frames.

For this test, we generated attacked versions of each uncompressed watermarked test video, and each compressed watermarked test video by uniformly randomly selecting frames from each video and removing them. We generated attacked versions with five, ten, fifteen, and twenty percent of the frames in each test video removed. We input the attacked videos into the decoder and calculated the BER of the decoded messages produced under decoder synchronisation.

Figure 5.12 shows the average BER vs. the percentage of frames dropped for the C_b and C_r channels of the uncompressed watermarked videos. The BER of the C_b channel is zero at a frame drop rate of five percent, and increases with increasing drop rate to 0.17 at a drop rate of twenty percent. The BER of the C_r channel is also zero at a drop rate of five percent, and increases to 0.13 at a drop rate of twenty percent.

Figure 5.13 shows the result for the compressed watermarked videos. The BER for both colour channels is higher at each drop rate than for the uncompressed test videos, and decoder synchronisation performs poorly for both colour channels. The BER at a drop rate of five percent is 0.15 and 0.22 for the C_b and C_r channels, respectively. The high BER may be due to the combined effect of the video compression and the frame dropping resulting in too much noise for the decoder to synchronise properly. The robustness of the scheme against frame dropping may possibly be improved by further reducing the temporal resolution of the watermark signal.

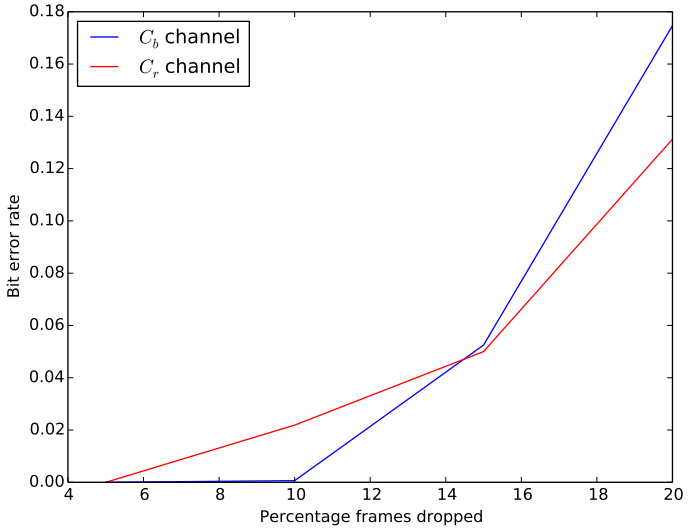


Figure 5.12: Average bit error rate vs. percentage of frames randomly deleted from the C_b and C_r channels of the uncompressed watermarked videos.

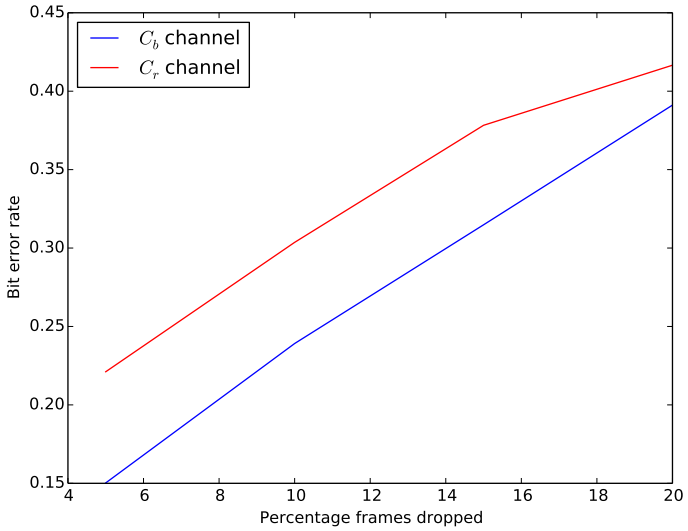


Figure 5.13: Average bit error rate vs. percentage of frames randomly deleted from the C_b and C_r channels of the compressed watermarked videos.

5.1.10 Gain and Noise Attack

The objective of this test is to evaluate the robustness of the digital watermarking technique against scaling of the pixel values of the C_b and C_r colour channels by a constant factor α, as well as the addition of noise to the scaled signal. In Section 2.1.7 on page 16 it was shown that scaling the amplitude of a QIM-based watermark signal is particularly effective at inducing decoding errors. In our watermarking scheme, the decoder estimates the factor by which the pixel values of the watermarked video signal have been scaled, and adjusts the quantiser step size used for decoding accordingly. An attacker may try to inhibit the decoder's ability to estimate the applied gain factor by adding noise to the gain-attacked video. The ability of the decoder to compensate for the applied gain determines the robustness of the watermarking scheme to gain attacks.

We first tested that the scheme is capable of decoding watermark messages from uncompressed watermarked videos that have been subjected to constant amplitude scaling and the addition of Gaussian noise. We generated five attacked versions of each uncompressed watermarked test video by scaling the pixel values of the C_b and C_r colour channels by gain factors of 0.85, 0.90, 0.95, 1.05, 1.10, 1.15. We then added Gaussian noise with standard deviations of 8, 16, 24, and 32, and input each gain and noise-attacked video into the decoder. Under both manual and decoder synchronisation, the average BER was zero for all gains and noise levels tested. This result was expected since in Section 5.1.5, the addition of Gaussian noise was found to be ineffective at inducing bit errors in the decoded watermark messages. The uncompressed gain-attacked videos still contained strongly quantised STDM signals on which the decoder could synchronise.

Because of the impracticality and ineffectiveness of adding Gaussian noise to the videos, when testing the compressed watermarked videos, we assumed that the attacker uses lossy video compression to induce the noise. As the bit rate to which a video is compressed increases, the amount of noise added to the video signal increases. We generated five attacked versions of each compressed watermarked test video by scaling the pixel values of the C_b and C_r colour channels by gain factors of 0.85, 0.90, 0.95, 1.05, 1.10, 1.15. We then compressed each of these attacked videos to bit rates of 30, 25, 20, 15, 10, and 5 Mbps to add increasing levels of noise to the attacked videos. We input the gain and noise-attacked videos into the decoder and calculated the BER for the manually synchronised and decoder synchronised decoded messages.

For this test, we expected that the results would be similar to those of the lossy compression test in Section 5.1.6 since the noise caused by the video compression should be similar, and, as discussed in Section 2.1.7 on page 16, constant gain scaling only changes the effective quantiser step size of the STDM signal. This means that the watermark message should not be corrupted by the pixel value scaling and, if the decoder can estimate α correctly, then it

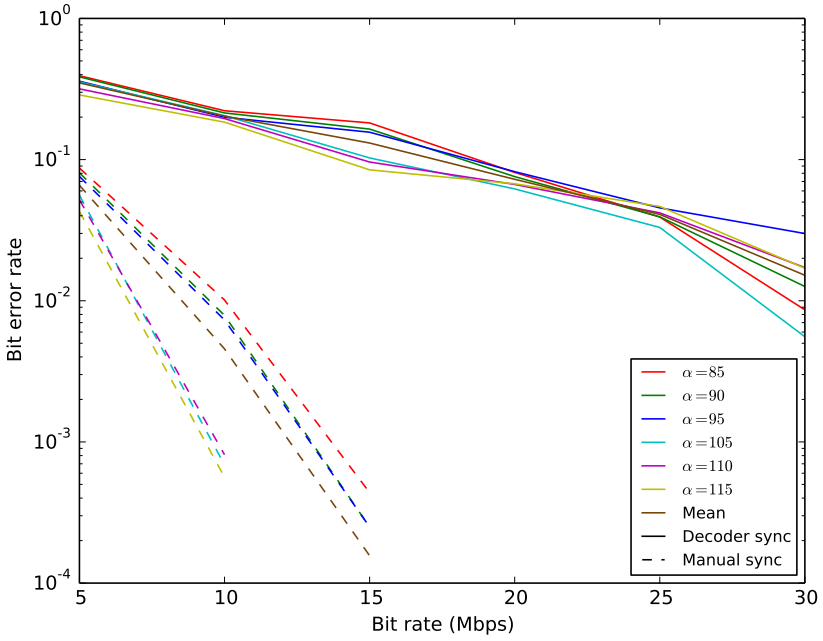


Figure 5.14: Average bit error rate vs. compressed bit rate of the C_b channel when subjected to amplitude scaling by a constant gain factor α.

should be able to decode the message correctly. However, one possible source of error is clipping of the scaled pixel values to lie within the range $\{0, \ldots, 255\}$ to be stored as byte values in the digital video file. This may result in some of the STDM values not being perfectly scaled by α.

Figure 5.14 and Figure 5.15 show the average BER at each compressed video bit rate for the C_b and C_r channels, respectively. For the C_b channel, under manual synchronisation, the mean BER over all values of α is zero at compressed bit rates above 15 Mbps, and increases with decreasing bit rate to 0.07 at a bit rate of 5 Mbps. The mean BER under decoder synchronisation is 0.02 at 30 Mbps, and increases with decreasing bit rate to 0.35 at 5 Mbps. For the C_r channel, under manual synchronisation, the mean BER over all values of α is zero at 30 Mbps, and increases with decreasing bit rate to 0.14 at a bit rate of 5 Mbps. The mean BER under decoder synchronisation is 0.06 at 30 Mbps, and increases with decreasing bit rate to 0.40 at 5 Mbps.

The manual synchronisation results were similar to the results of the lossy compression test in Section 5.1.6, but BER under decoder synchronisation was greater for this test than for the lossy compression test, with errors already occurring at a bit rate of 30 Mbps. The high BER under decoder synchronisation is likely due to the noise induced by the recompression of the gain-attacked, compressed watermarked videos being too great to allow the decoder to synchronise properly on the STDM signal.

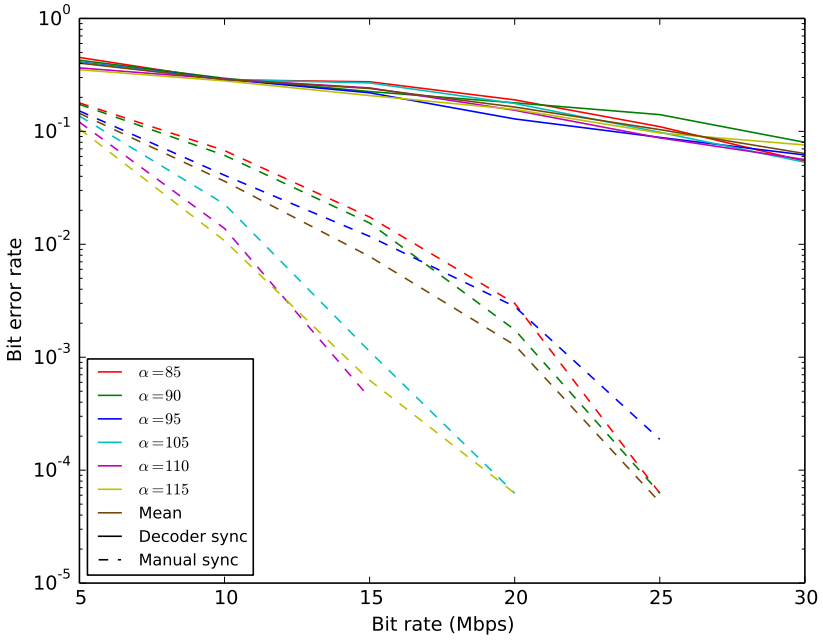


Figure 5.15: Average bit error rate vs. compressed bit rate of the C_r channel when subjected to amplitude scaling by a constant gain factor α.

5.1.11 Recompression

A compressed watermarked video may be recompressed using a different compression algorithm. This may be done in order to change the format of a digital video to suit a particular display device, or it may be done by a pirate with the intention of destroying the embedded watermark signal. The objective of this test is to determine the robustness of the watermarking scheme to recompression of compressed watermarked videos.

For this test, we recompressed the compressed watermarked test videos, using the MPEG4 video compression algorithm. We compressed the test videos to bit rates of 30 Mbps, 20 Mbps, 15 Mbps, and 10 Mbps. Bit rates lower than 10 Mbps were not achievable using MPEG4 compression for the resolution and frame rate of the test videos. We input the attacked videos into the decoder and calculated the BER for the decoded messages produced under manual and decoder synchronisation.

Figure 5.16 shows the BER vs. bit rate of the recompressed videos. Under manual synchronisation, the BER for the C_b channel increases with decreasing bit rate from 0.03 at 30 Mbps to 0.12 at 10 Mbps, and the BER for the C_r channel increases from 0.06 at 30 Mbps to 0.17 at 10 Mbps. The robustness of the scheme under decoder synchronisation is poor, with a BER of 0.19 at 30 Mbps for the C_b channel, and a BER of 0.30 at 30 Mbps for the C_r channel. The noise induced by the compression and recompression of the watermarked

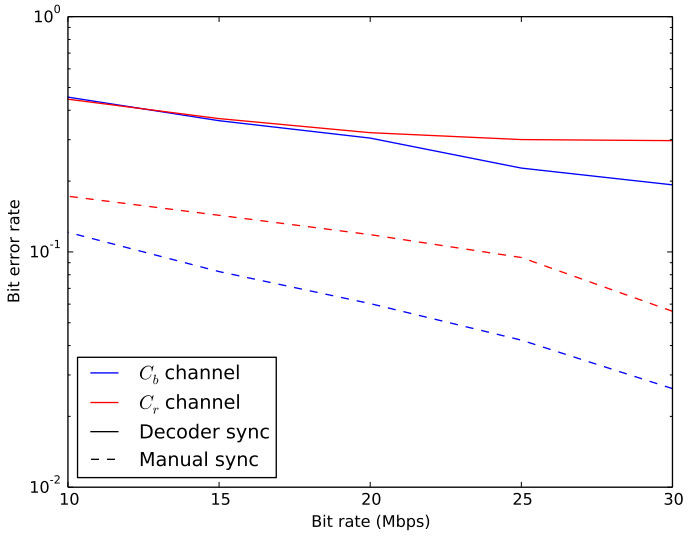


Figure 5.16: Average bit error rate of the C_b and C_r channels vs. bit rate of the videos recompressed using MPEG4 compression.

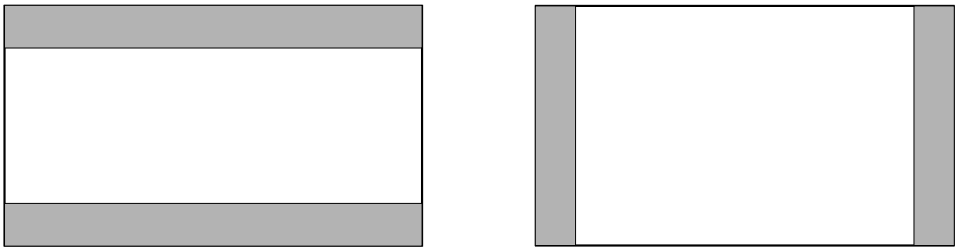
video is too great for the decoder to synchronise properly, and even under manual synchronisation there are decoding errors even at the highest bit rate tested. The watermarking parameters may need to be modified to improve the robustness of a production version of this prototype scheme against re-compression.

5.1.12 Horizontal Cropping

The digital watermarking scheme is not specifically designed to be robust against cropping of watermarked videos. However, determining the inherent robustness of the scheme is useful for future work that focusses on ways of modifying the scheme to be robust against cropping. In this thesis, we considered two kinds of cropping, which we refer to as horizontal and vertical cropping. Horizontal cropping is the removal of rows of pixels from the top and bottom of each frame, and vertical cropping is the removal of columns of pixels from the left and right of each frame. Figure 5.17 illustrates these two cropping techniques.

The objective of this test is to determine the robustness of the scheme to horizontal cropping. We first generated cropped versions of each uncompressed watermarked video by removing 8, 16, 24, and 32 rows of pixels from both the top and bottom of each frame. We input the cropped videos into the decoder and calculated the BER under manual and decoder synchronisation. We then repeated the test using the compressed watermarked test videos.

We expected that the BER under decoder synchronisation would be very



Horizontal cropping Vertical cropping

Figure 5.17: Illustration of horizontal and vertical cropping. The shaded regions are the bands from which pixels are removed by the corresponding cropping operation.

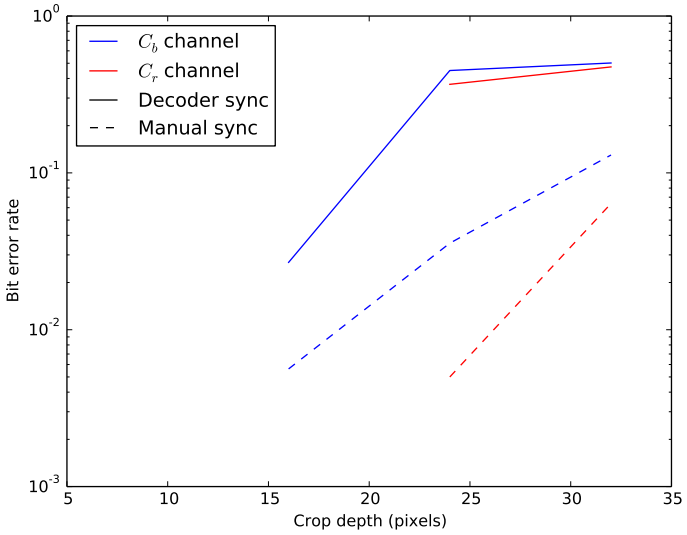


Figure 5.18: Average bit error rate of the C_b and C_r channels vs. number of rows of pixels removed from the top and bottom of the uncompressed watermarked videos by horizontal cropping.

high, even for a small number of pixels removed. This is because the synchronisation system is not designed to detect cropping. We also expected that the BER under manual synchronisation would be high due to the misalignment of the STDM chip signals with the watermark signals in the cropped videos.

Figure 5.18 shows the average BER of the C_b and C_r channels vs. the number of rows of pixels removed from the uncompressed watermarked videos. The BER of the C_b channel is zero under both manual and decoder synchronisation for a crop depth of eight pixels. As expected, the BER under decoder synchronisation is higher than the BER under manual synchronisation, increasing with crop depth to 0.50 at a crop depth of 32 pixels. The BER of the C_r channel is zero under both manual and decoder synchronisation for crop depths of eight

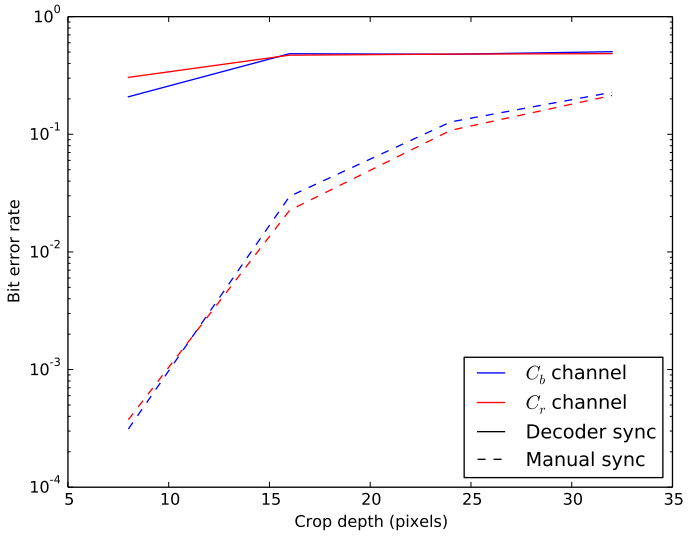


Figure 5.19: Average bit error rate of the C_b and C_r channels vs. number of rows of pixels removed from the top and bottom of the compressed watermarked videos by horizontal cropping.

and sixteen pixels. Again, the BER under decoder synchronisation is higher than the BER under manual synchronisation, increasing with crop depth to 0.47 at a crop depth of 32 pixels.

Figure 5.19 shows the average BER for the C_b and C_r channels vs. the number of rows of pixels removed from the compressed watermarked videos. As the number of cropped pixels increases, the BER under manual synchronisation rises from 3.13×10^{-4} to 0.22 for the C_b channel, and from 3.75×10^{-4} to 0.21 for the C_r channel. As expected, the BER under decoder synchronisation is much higher than under manual synchronisation, rising from 0.21 to 0.50 for the C_b channel, and from 0.30 to 0.49 for the C_r channel.

5.1.13 Vertical Cropping

The objective of this test is to determine the robustness of the scheme to vertical cropping. We first generated cropped versions of each uncompressed watermarked video by removing 8, 16, 24, and 32 columns of pixels from both the left and right of each frame. We input the cropped videos into the decoder and calculated the BER under manual and decoder synchronisation. We then repeated the test using the compressed watermarked test videos.

Figure 5.20 shows the average BER of the C_b and C_r channels vs. the number of columns of pixels removed from the uncompressed watermarked videos. The results are similar to the test above. The BER of both channels under manual and decoder synchronisation is zero for a crop depth of eight

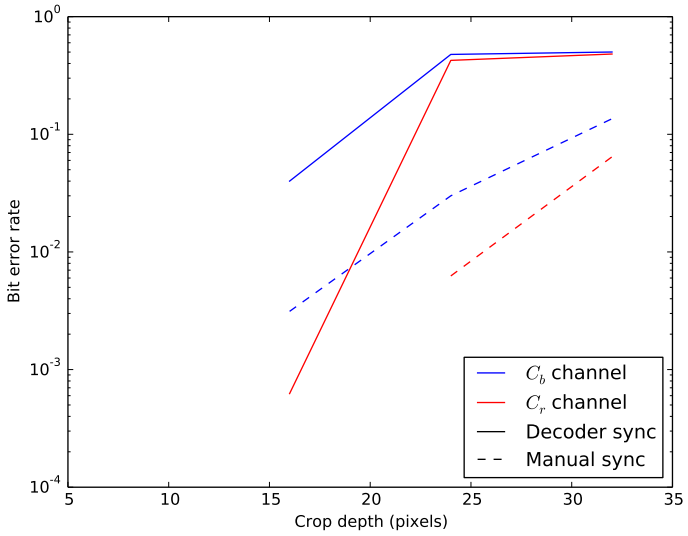


Figure 5.20: Average bit error rate of the C_b and C_r channels vs. number of columns of pixels removed from the left and right of the uncompressed watermarked videos by vertical cropping.

pixels. Again, the BER under decoder synchronisation is higher than the BER under manual synchronisation. At a crop depth of 32 pixels, the BER under decoder synchronisation is 0.50 for the C_b channel, and 0.48 for the C_r channel.

Figure 5.21 shows the average BER of the C_b and C_r channels vs. the number of columns of pixels removed from the compressed watermarked videos by vertical cropping. As the number of cropped pixels increases, the BER under manual synchronisation rises from 2.5×10^{-4} to 0.23 for the C_b channel, and from 6.25×10^{-5} to 0.21 for the C_r channel. Once again, the BER under decoder synchronisation is much higher than under manual synchronisation, rising from 0.19 to 0.50 for the C_b channel, and from 0.28 to 0.47 for the C_r channel.

The results of this test and the test above show that, while the decoder synchronisation performs poorly, the manually synchronised watermark signal is still somewhat robust under the cropping parameters tested. Additionally, for the uncompressed test videos, the BER under decoder synchronisation was still zero for the lowest crop depth tested. The decoder could be modified in the future to allow it to synchronise more effectively with watermark signals in cropped videos. A decoder modified in this way could theoretically decode watermark messages with BER similar to the BER under manual synchronisation.

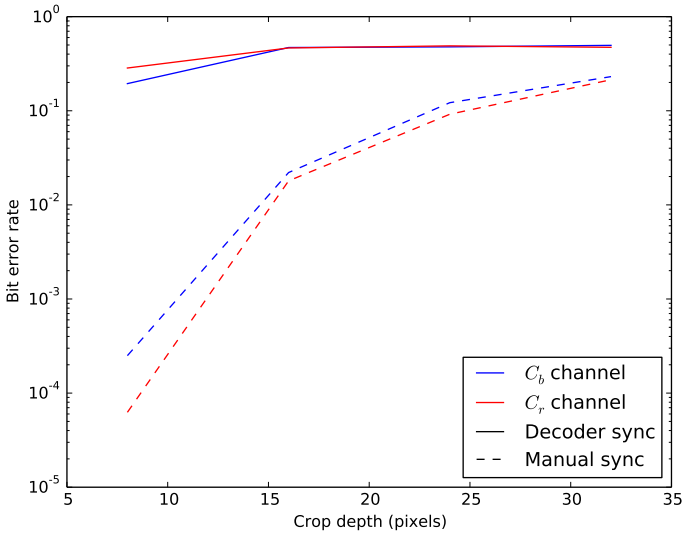


Figure 5.21: Average bit error rate of the C_b and C_r channels vs. number of columns of pixels removed from the left and right of the compressed watermarked videos by vertical cropping.

5.1.14 Discussion

In this section we tested the robustness of the watermarking scheme against several attacks and video processing operations. We showed that when the watermarked videos remain uncompressed, the watermarking scheme works as expected and is robust against all of the attacks tested. We also showed that when the watermarked videos are compressed, and the attacks are not too severe, a low watermark message BER can be achieved under manual synchronisation, which means that the watermark signals are capable of remaining mostly uncorrupted in the attacked compressed videos.

For the tests carried out on compressed watermarked videos, the BER under decoder synchronisation was generally much higher than the BER under manual synchronisation. This indicates that the STDM signals in the attacked videos correctly encoded most of the watermark message data, but the signals were too noisy to be detected during decoder synchronisation. Video compression algorithms are designed to remove information from digital video signals that is barely noticeable to the viewer [6], and since the watermark signal is intended not to be noticeable, compression can severely degrade the STDM watermark signals. The losses incurred in resampling and recompressing the attacked videos may also have added noise to the STDM signal. The robustness of decoder synchronisation with compressed watermarked videos could be improved by increasing the quantiser step size (which would lower the raw bit rate of the watermark signal), or by incorporating other means of synchronisation into the watermark message bits. For example, some of the bits encoded

into each block can be used to assist the decoder in identifying that block.

For most of the tests carried out on the compressed versions of the watermarked test videos, watermarks embedded in the C_r channel incurred a greater BER than those embedded in the C_b channel. In Section 5.1.2 it was shown that the average variance of the C_b channels of the unmarked test videos is more than twice as high as that of the C_r channels. The C_b channels of the test videos may be less affected by lossy compression than the C_r channels, due to the video compression algorithm preserving more of the C_b signal, whose power contributes more to the video signal, and discarding more of C_r channel, whose power contributes less. In a more developed watermarking scheme, the watermark encoder can be designed to detect how suitable a particular colour channel is for watermarking by calculating the variance of each channel. The parameters of the watermark embedded into each channel can then be adjusted according to the properties of the channel.

5.2 The Dynamic Traitor Tracing Scheme

In this section we compare the performance of our dynamic traitor tracing scheme to that of the FT scheme. Our scheme uses a delay-tolerant search tree algorithm, described in Section 3.2.1 on page 39, to partition the users in U, and the dynamic Tardos code construction, described in Section 3.2.3 on page 44, to generate watermark messages for each segment variant. The FT scheme uses the search tree algorithm described in Section 2.2.6 on page 26. In order to compare the schemes more fairly, we use the dynamic Tardos code construction, described in Section 2.2.5 on page 24, to generate watermark messages for video segment watermarking in the FT scheme. This ensures that the optimised Tardos constants are the same for our scheme and for the FT scheme.

5.2.1 Methodology

We tested the dynamic traitor tracing schemes by running computer simulations of broadcasts of watermarked content to a set of users. We ran 10 computer simulations of each collusion attack scenario and averaged the results. For each simulation, we randomly selected a set of users to be group of colluding pirates who carry out a collusion attack on the watermarked content, and rebroadcast the content. For these simulations we assumed that the underlying digital watermark modulation scheme complies with the marking assumption, discussed in Section 2.2.1.2 on page 19, and that the correct watermark message bits can be decoded from the received pirate copies. We therefore did not physically watermark video content, but we represented the watermarked content assigned to each user by the watermark message itself.

The intended advantage of our scheme over the FT scheme is that the time required by our scheme to disconnect all colluding pirates is shorter than the standard scheme when feedback from the pirate redistribution is delayed. Since our computer simulations did not run in real-time using actual broadcast video, we could not directly measure the time taken for each scheme to trace and disconnect all pirates. Instead, we made use of the assumption we made in Section 2.2.7 on page 28 that the watermarking bit rate is constant at r_{wm} bits per second. Using this assumption, the time taken to trace all pirates is proportional to the number of watermark bits transmitted to each user, so we use this as our metric for the time required to trace all pirates. We also express the feedback delay between segment transmission and the receipt of pirate segments in terms of watermark message bits.

In addition to measuring the number of watermark message bits transmitted to each user, for each simulation, we also measured the total number of watermark message bits transmitted by the scheme. We calculated the total number of watermark bits transmitted during a video segment as the product of the number of variants of the video segment and the watermark message length for that segment. The total number of bits transmitted was then calculated as the sum of the total bits transmitted during every segment.

5.2.2 Random Bit Selection

Section 2.2.1.2 on page 19 discussed the bit selection collusion attack. One strategy the pirates can use to select the watermark message bits with which they construct the pirate rebroadcast is to randomly select between the watermark bits present in all of the videos available to them. The objective of this test is to evaluate the performance of our dynamic traitor tracing scheme when the pirates use this random bit selection strategy.

The simulated pirates generated the collusion-attacked watermark messages as follows. For each bit in the collusion-attacked watermark message, where the pirates had access to both the '1' and '0' version, the value of the bit retransmitted by the pirates was independently randomly chosen to be '1' with a probability of 0.5. For each bit where the pirates had access to only one version, that version was retransmitted, in accordance with the marking assumption. The traitor tracing scheme received feedback from the simulated pirate rebroadcast after a time delay, t_{delay}, corresponding to a number of watermark message bits, b_{delay}. We tested the scheme for b_{delay} values of 0, 2000, 4000, 6000, 8000, and 10000 bits.

Figure 5.22 shows the number of watermark message bits transmitted to each user by our scheme and by the FT scheme in order to trace all pirates vs. the amount of feedback delay from the pirate redistribution, when the bit selection collusion attack is used. For this test, the total number of users was 10^6. From the results we can see that our scheme requires fewer equivalent watermark bits to be transmitted to each user when feedback is delayed. This

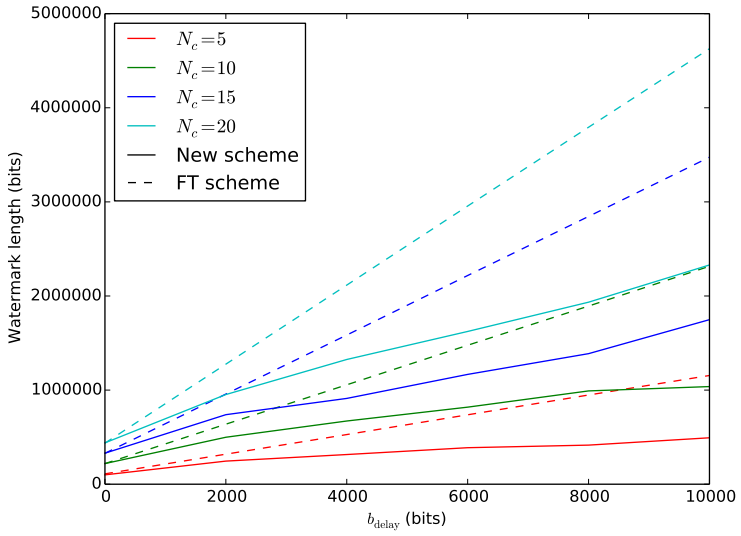


Figure 5.22: Number of watermark message bits transmitted to each user by each scheme vs. feedback delay, b_{delay}, when the random bit selection collusion strategy is used. The total number of users, N_u, is 10^6, and the number of colluding pirates is N_c.

means that our scheme requires less broadcast time to trace all of the pirates than the FT scheme.

Figure 5.23 shows the total number of bits transmitted by our scheme and by the FT scheme. This result shows that the total number of bits transmitted by our scheme is more than that of the FT scheme. This is due to the fact that our scheme uses more watermarked variants per segment than the FT scheme.

5.2.3 Scapegoat Strategy

Another strategy the pirates may employ is to rebroadcast the video segments received by one of the members of C until that member is disconnected, and then to start rebroadcasting the video received by the next member, and so on. In [16] this strategy is referred to as the *scapegoat strategy* since the pirate whose watermarked content is rebroadcast until disconnection acts as a scapegoat for the others.

For this test, the simulated pirates generated the collusion-attacked watermark messages as follows. One pirate in C was selected at random to be the scapegoat. For each bit in the collusion-attacked watermark message, the bit retransmitted by the pirates was chosen to be the watermark bit received by the scapegoat pirate. When the scapegoat was disconnected, another scapegoat was chosen at random, and the collusion attack continued until all pirates were disconnected. The traitor tracing scheme received feedback from the sim-

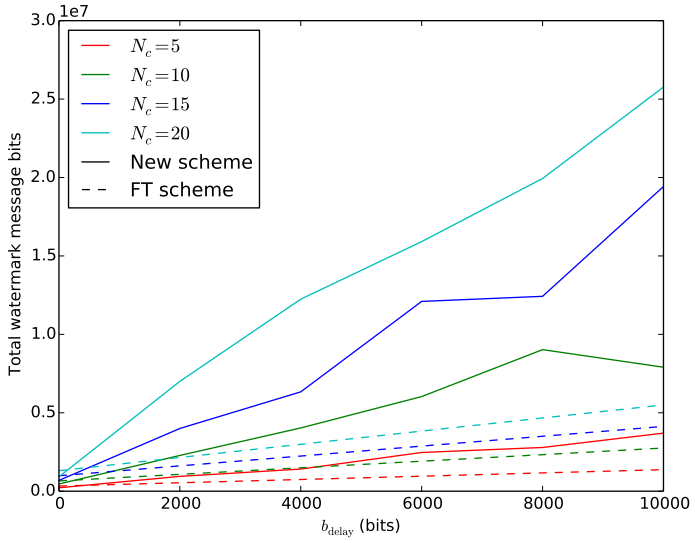


Figure 5.23: Total number of watermark message bits transmitted by each scheme vs. feedback delay, b_{delay}, when the random bit selection collusion strategy is used. The total number of users, N_u, is 10^6, and the number of colluding pirates is N_c.

ulated pirate rebroadcast after a time delay, t_{delay}, corresponding to a number of watermark message bits, b_{delay}. We tested the scheme for b_{delay} values of 0, 2000, 4000, 6000, 8000, and 10000 bits.

Figure 5.24 shows the number of watermark message bits transmitted to each user by our scheme and by the FT scheme in order to trace all pirates vs. the amount of feedback delay from the pirate redistribution, when the scapegoat strategy is used. For this test, the total number of users was 10^6. Our scheme once again requires fewer equivalent watermark bits to be transmitted to each user, and therefore, less broadcast time, than the FT scheme, when feedback is delayed. Figure 5.25 shows the total number of bits transmitted by our scheme and by the FT scheme. As in the previous test, the total number of bits transmitted by our scheme is more than that of the FT scheme, due to the extra watermarked variants used.

5.2.4 Discussion

In broadcast digital distribution systems, where it is impractical to transmit unique watermark messages to each user, partitioning the users into sets using the FT scheme can allow a set of N_c colluding pirates to be traced using a maximum of $2N_c+1$ simultaneous unique watermark messages [18]. The results of the tests in this section show that when using Tardos codes to construct watermark messages, the traitor tracing scheme we implemented provides a trade-off between the time taken for the scheme to disconnect all colluding

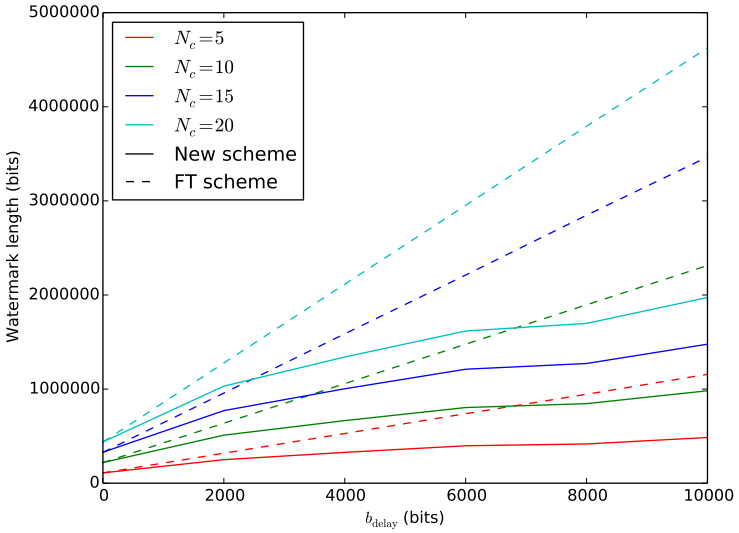


Figure 5.24: Number of watermark message bits transmitted to each user by each scheme vs. feedback delay, b_{delay}, when the scapegoat collusion strategy is used. The total number of users, N_u, is 10^6, and the number of colluding pirates is N_c.

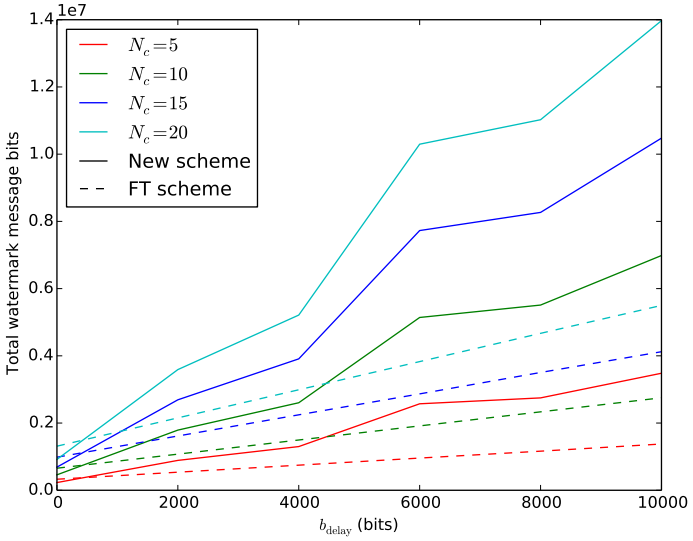


Figure 5.25: Total number of watermark message bits transmitted by each scheme vs. feedback delay, b_{delay}, when the scapegoat collusion strategy is used. The total number of users, N_u, is 10^6, and the number of colluding pirates is N_c.

pirates, and the total watermarking bandwidth used, when compared to the FT scheme. If a broadcast scheme that uses dynamic traitor tracing is capable of supporting the extra total bandwidth, then our traitor tracing scheme can be used to disconnect all colluding pirates more quickly than the FT scheme.

Chapter 6

Conclusion

In this thesis we designed and implemented a prototype digital watermarking scheme for embedding hidden binary message signals in uncompressed digital video. We tested the robustness of the scheme when watermarked videos are modified by video processing techniques, as well as by intentional attacks on the scheme. The scheme was shown to be robust against the attacks tested when the watermarked and attacked videos remained uncompressed. However, when the attacks were carried out on the compressed versions of watermark videos, decoder synchronisation performed poorly, and bit error rates were high. In order to develop the prototype scheme into a production watermarking system, robustness to attack when using compressed videos needs to be increased.

In order to protect digitally watermarked broadcast video from collusion attack we also designed and implemented a modified version of Fiat and Tassa's dynamic traitor tracing algorithm for generating watermark messages. The test results show that the new scheme is capable of tracing and disconnecting all colluding pirates more quickly than the FT scheme when feedback from the pirate rebroadcast is delayed, at the expense of using more watermarked segment variants and, therefore, a greater total watermark message bandwidth.

6.1 Future Work

If the prototype scheme implemented in this thesis is to be developed into a production digital watermarking scheme, there are some features that still need to be added or improved. Below are some suggestions for improving and extending the scheme in future work.

6.1.1 Optimisation of Watermark Modulation Parameters

The parameters chosen in Section 4.1 on page 50 for the watermark modulation scheme were somewhat arbitrary and therefore do not necessarily provide an optimal trade-off between fidelity, security, and robustness. Parameters such as the temporal and spatial resolution of the chip signals and the number of watermark message bits encoded into each block could be optimised to provide the best trade-off between robustness, fidelity, and data payload. The scheme can also be modified so that the watermarking parameters can vary between blocks in order to best suit the characteristics of each block.

6.1.2 Improvement of Robustness to Cropping

The decoder synchronisation of our watermarking scheme does not take frame cropping into account, and therefore the BER under decoder synchronisation is high even for minor cropping attacks. The BER under manual synchronisation is low for the cropping attacks tested, which means that the system could be modified to synchronise with a cropped video signal, and achieve a low BER. One possible solution is to let the watermark signal encoded in each frame be spatially smaller than the video frame itself and centred within the frame so that vertical and horizontal bands of pixels around the edge of the frame contain no watermark signal. Any cropping operation that only removes pixels from the un-watermarked bands would not damage the watermark signal. The decoder would still require a method of achieving spatial synchronisation with the watermark signal, since cropping may change the position and relative size of the watermarked region within each frame. A method similar to the one used by our decoder to achieve temporal synchronisation might be appropriate.

6.1.3 Improvement of Robustness to Compression

Our watermark modulation scheme is designed to operate on uncompressed digital video. This has the advantage that the scheme is not dependent on any particular video compression algorithm or compressed video format. However, knowledge of how video compression algorithms work could be used to improve the robustness of the scheme against video compression in general, or against particular compression algorithms in specific cases where this is sufficient. Our scheme, or at least the parameters thereof, could be modified so that the watermark signal is strong in regions of the host video signal that are least affected by compression, and weak or not present in regions that are heavily affected by compression. Another reason to make the scheme "aware" of compression algorithms is to prevent the watermark signal from significantly increasing the size of the watermarked video file over that of the unmarked host video file.

6.1.4 Incorporation of a Perceptual Model

In this thesis we focussed mainly on the robustness of our digital watermarking scheme. While we measured the level of distortion induced in the videos by the watermarks in terms of DWR, we did not consider the perceptibility of this distortion to a viewer. A perceptual model can be used as a means of determining the level of noticeable distortion induced in a host video, and to improve the simple method used by our prototype watermark encoder to distribute the watermark signal within the host.

The random choice of Ω'_{k_i}, the set of coordinates for which the watermark signal is non-zero, made in Section 3.1.4 on page 33, can be substituted for a choice that selects locations in v_k for which the added watermark signal will be least noticeable to a user. The method we chose for upsampling the low resolution watermark signal, $\tilde{w}_k^{(u)}[x, y, t]$, to give the high resolution signal, $w_k^{(u)}[x, y, t]$, is also very simple and does not take advantage of knowledge of the host signal. In a more developed version of our watermarking scheme, the high resolution watermark signal can be generated from the low resolution signal in a way that blends in better with the host signal. This would improve the fidelity of the watermarking scheme.

6.1.5 Digital Layer

In this thesis we implemented a digital watermark modulation scheme that can embed an arbitrary message into a host signal, and decode those bits from the (possibly distorted) watermarked signal. A logical next step would be to develop a digital communication protocol which uses the available watermark bits in an optimal way to provide robustness against bit errors, as well as a large enough data payload to carry useful user-identifying information. The protocol may include forward error correction, and template sequences that can assist in decoder synchronisation. Some type of packet-based system may enable the decoder to reconstruct messages from out-of-order watermark blocks, and may allow packets to be repeated to allow successful decoding when some packets are lost.

The decoding algorithm described in Section 3.1.8 on page 39 uses hard-decision decoding for each bit. Since the determination of each received bit value relies on a decision score, forward error correction using soft-decision decoding can be implemented. In [56] it is shown that soft-decision decoding can improve the probability of successful watermark decoding. The quantisation metric, β, generated by the gain estimator described in Section 3.1.7 on page 37, can also be used as a parameter when determining the most likely estimate of the watermark message signal.

6.1.6 Implementation in Broadcast Systems

Our prototype implementation was developed in Python, and tested using computer simulation. If the scheme is to be implemented in pay-TV broadcast systems then some of the software may need to be rewritten to run on the set-top box hardware used by the pay-TV system. Another issue is the method by which the common video content and unique watermarks will be transmitted to the users. Broadcast encryption schemes, such as the one presented in [48], allow authorised users to access a broadcast, while denying access to unauthorised users. However, these schemes are not designed to deliver unique copies to each authorised user.

Adapting broadcast encryption schemes to include multiple watermarked copies by simply broadcasting each watermarked copy may require unacceptably large bandwidth. Broadcasting the original content and relying on the set-top box itself to embed the watermarks would require less extra bandwidth, but the CP would have to take measures to secure the unmarked content so that only the watermarked versions generated by set-top boxes can be accessed. This may place unrealistic security and tamper-resistance requirements on set-top box hardware.

A digital watermarking scheme for internet multicast is proposed in [57], and may be modified for a broadcast setting. In this scheme two differently encrypted versions of the original content are generated and distributed to users. The first version is digitally watermarked with a message consisting of only 1's, and the second is watermarked with a message containing only 0's. The parts of each version corresponding to individual bits are encrypted using different keys. For each watermark message bit, each set-top box can only decrypt the corresponding part from one of the watermarked versions. The set-top box combines the parts it can decrypt into a uniquely watermarked video. This scheme requires that no more than two versions of the same content be broadcast, and the unmarked content is never exposed, but additional bandwidth is required to make the correct decryption keys available to each set-top box. In this case a traitor tracing scheme such as the one implemented in this thesis can be used to generate the sets of decryption keys.

List of References

[1] J. A. Bloom, I. J. Cox, T. Kalker, J.-P. Linnartz, M. L. Miller, and C. B. S. Traw, "Copy protection for DVD video," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1267–1276, 1999.

[2] M. Maes, T. Kalker, J.-P. Linnartz, J. Talstra, F. Depovere, and J. Haitsma, "Digital watermarking for DVD video copy protection," *Signal Processing Magazine, IEEE*, vol. 17, no. 5, pp. 47–57, 2000.

[3] Q.-M. Ge, Z.-M. Lu, and X.-M. Niu, "Oblivious video watermarking scheme with adaptive embedding mechanism," in *Machine Learning and Cybernetics, 2003 International Conference on*, vol. 5. IEEE, 2003, pp. 2876–2881.

[4] J. Zhang, J. Li, and L. Zhang, "Video watermark technique in motion vector," in *Computer Graphics and Image Processing, 2001 Proceedings of XIV Brazilian Symposium on*. IEEE, 2001, pp. 179–182.

[5] C.-Y. Lin, M. Wu, J. A. Bloom, I. J. Cox, M. L. Miller, and Y. M. Lui, "Rotation, scale, and translation resilient watermarking for images," *Image Processing, IEEE Transactions on*, vol. 10, no. 5, pp. 767–782, 2001.

[6] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker, *Digital Watermarking and Steganography*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[7] R. H. van Huyssteen, "Comparative evaluation of video watermarking techniques in the uncompressed domain," Master's thesis, Stellenbosch: Stellenbosch University, 2012.

[8] I. J. Cox and J.-P. Linnartz, "Some general methods for tampering with watermarks," *Selected Areas in Communications, IEEE Journal on*, vol. 16, no. 4, pp. 587–593, 1998.

[9] G. Simmons, *Contemporary Cryptology: The Science of Information Integrity*. Wiley, 1999. [Online]. Available: http://books.google.co.za/books?id=gA1EPgAACAAJ

[10] M. L. Miller, I. J. Cox, J.-P. M. Linnartz, and T. Kalker, "A review of watermarking principles and practices," *Digital Signal Processing for Multimedia Systems*, pp. 461–485, 1999.

[11] Q. Li and I. J. Cox, "Using perceptual models to improve fidelity and provide resistance to valumetric scaling for quantization index modulation watermarking," *Information Forensics and Security, IEEE Transactions on*, vol. 2, no. 2, pp. 127–139, 2007.

[12] F. Pérez-González, C. Mosquera, M. Barni, and A. Abrardo, "Rational dither modulation: A high-rate data-hiding method invariant to gain attacks," *Signal Processing, IEEE Transactions on*, vol. 53, no. 10, pp. 3960–3975, 2005.

[13] F. Hartung and B. Girod, "Watermarking of uncompressed and compressed video," *Signal Processing, IEEE Transactions on*, vol. 66, no. 3, pp. 283–301, 1998.

[14] X. Niu, M. Schmucker, and C. Busch, "Video watermarking resistance to rotation, scaling, and translation," in *Electronic Imaging 2002*. International Society for Optics and Photonics, 2002, pp. 512–519.

[15] F. Deguillaume, G. Csurka, J. J. O'Ruanaidh, and T. Pun, "Robust 3D DFT video watermarking," in *Electronic Imaging'99*. International Society for Optics and Photonics, 1999, pp. 113–124.

[16] T. Laarhoven, J. Doumen, P. Roelse, B. Skoric, and B. de Weger, "Dynamic Tardos traitor tracing schemes," *Information Theory, IEEE Transactions on*, vol. 59, no. 7, pp. 4230–4242, 2013.

[17] G. Tardos, "Optimal probabilistic fingerprint codes," in *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*. ACM, 2003, pp. 116–125.

[18] A. Fiat and T. Tassa, "Dynamic traitor tracing," in *Advances in Cryptology-CRYPTO'99*. Springer, 1999, pp. 354–371.

[19] P. Roelse, "Dynamic subtree tracing and its application in pay-TV systems," *International Journal of Information Security*, vol. 10, no. 3, pp. 173–187, 2011.

[20] Y. Song and T. Tan, "Comparison of four different digital watermarking techniques," in *Signal Processing Proceedings, 2000. WCCC-ICSP 2000. 5th International Conference on*, vol. 2. IEEE, 2000, pp. 946–950.

[21] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for images, audio and video," in *Image Processing,*

1996. Proceedings., International Conference on, vol. 3. IEEE, 1996, pp. 243–246.

[22] C.-T. Hsu and J.-L. Wu, "DCT-based watermarking for video," *Consumer Electronics, IEEE Transactions on*, vol. 44, no. 1, pp. 206–216, 1998.

[23] M. D. Swanson, B. Zhu, and A. H. Tewfik, "Data hiding for video-in-video," in *Image Processing, 1997. Proceedings., International Conference on*, vol. 2. IEEE, 1997, pp. 676–679.

[24] P.-W. Chan and M. R. Lyu, "A DWT-based digital video watermarking scheme with error correcting code," in *Information and Communications Security*. Springer, 2003, pp. 202–213.

[25] H. Liu, N. Chen, J. Huang, X. Huang, and Y. Q. Shi, "A robust DWT-based video watermarking algorithm," in *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, vol. 3. IEEE, 2002, pp. 631–634.

[26] B. Chor, A. Fiat, and M. Naor, "Tracing traitors," in *Advances in Cryptology-CRYPTO'94*. Springer, 1994, pp. 257–270.

[27] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *Information Theory, IEEE Transactions on*, vol. 44, no. 5, pp. 1897–1905, 1998.

[28] T. Tassa, "Low bandwidth dynamic traitor tracing schemes," *Journal of Cryptology*, vol. 18, no. 2, pp. 167–183, 2005.

[29] B.-H. Cha and C.-C. Kuo, "Design of collusion-free hiding codes using MAI-free principle," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 2. IEEE, 2007, pp. 145–148.

[30] ——, "Design of multiuser collusion-free hiding codes with delayed embedding," in *Intelligent Information Hiding and Multimedia Signal Processing, 2007. IIHMSP 2007. Third International Conference on*, vol. 1. IEEE, 2007, pp. 379–382.

[31] H. Yagi, T. Matsushima, and S. Hirasawa, "New traceability codes against a generalized collusion attack for digital fingerprinting," in *Information Security Applications*. Springer, 2007, pp. 252–266.

[32] Y.-T. Lin, J.-L. Wu, and C.-H. Huang, "Concatenated construction of traceability codes for multimedia fingerprinting," *Optical Engineering*, vol. 46, no. 10, pp. 1–15, 2007.

[33] M. U. Celik, G. Sharma, and A. M. Tekalp, "Collusion-resilient fingerprinting using random prewarping," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 1. IEEE, 2003, pp. 831–835.

[34] W. Luh and D. Kundur, "New paradigms for effective multicasting and fingerprinting of entertainment media," *Communications Magazine, IEEE*, vol. 43, no. 6, pp. 77–84, 2005.

[35] K. Jack, *Video Demystified: A Handbook for the Digital Engineer*, 3rd ed., ser. Demystifying Technology Series. Elsevier, 2001. [Online]. Available: http://books.google.co.za/books?id=Q5ufQgAACAAJ

[36] B. Lathi and Z. Ding, *Modern Digital and Analog Communication Systems*, ser. The Oxford Series in Electrical and Computer Engineering Series. Oxford University Press, 2009. [Online]. Available: http://books.google.co.za/books?id=dltNPwAACAAJ

[37] F. Hartung, J. K. Su, and B. Girod, "Spread spectrum watermarking: Malicious attacks and counterattacks," *Security and Watermarking of Multimedia Contents, Proc SPIE*, vol. 3657, p. C1, 1999.

[38] A. Viterbi, *CDMA: Principles of Spread Spectrum Communication*, ser. Addison-Wesley Wireless Communications Series. Addison Wesley Publishing Company Incorporated, 1995. [Online]. Available: http://books.google.co.za/books?id=eSdvQgAACAAJ

[39] B. Chen and G. W. Wornell, "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *Information Theory, IEEE Transactions on*, vol. 47, no. 4, pp. 1423–1443, 2001.

[40] R. B. Wolfgang, C. I. Podilchuk, and E. J. Delp, "Perceptual watermarks for digital images and video," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1108–1126, 1999.

[41] I. J. Cox, M. L. Miller, and A. L. McKellips, "Watermarking as communications with side information," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1127–1141, 1999.

[42] C. E. Shannon, "Channels with side information at the transmitter," *IBM Journal of Research and Development*, vol. 2, no. 4, pp. 289–293, 1958.

[43] M. Costa, "Writing on dirty paper (corresp.)," *Information Theory, IEEE Transactions on*, vol. 29, no. 3, pp. 439–441, 1983.

[44] J. J. Eggers, R. Baeuml, and B. Girod, "Estimation of amplitude modifications before SCS watermark detection," in *Electronic Imaging 2002*. International Society for Optics and Photonics, 2002, pp. 387–398.

[45] I. D. Shterev, R. L. Lagendijk, and R. Heusdens, "Statistical amplitude scale estimation for quantization-based watermarking," in *Proceedings of SPIE*, vol. 5306, 2004, pp. 796–804.

[46] T. Laarhoven and B. de Weger, "Optimal symmetric Tardos traitor tracing schemes," *Designs, Codes and Cryptography*, vol. 71, no. 1, pp. 83–103, 2014.

[47] B. Škorić, S. Katzenbeisser, and M. U. Celik, "Symmetric Tardos fingerprinting codes for arbitrary alphabet sizes," *Designs, Codes and Cryptography*, vol. 46, no. 2, pp. 137–166, 2008.

[48] A. Fiat and M. Naor, "Broadcast encryption," in *Advances in Cryptology-CRYPTO'93*. Springer, 1994, pp. 480–491.

[49] Python Software Foundation, "Python Programming Language Official Website," http://www.python.org/, 2014.

[50] X. Cai, H. P. Langtangen, and H. Moe, "On the performance of the Python programming language for serial and parallel scientific computations," *Scientific Programming*, vol. 13, no. 1, pp. 31–56, 2005.

[51] "NumPy," http://www.numpy.org/, 2014.

[52] C. Basile, A. P. Cavallerano, M. S. Deiss, R. Keeler, J. S. Lim, W. C. Luplow, W. H. Paik, E. Petajan, R. Rast, and G. Reitmeier, "The US HDTV standard," *Spectrum, IEEE*, vol. 32, no. 4, pp. 36–45, 1995.

[53] libav, "Open source audio and video processing tools," http://www.libav.org/, 2014.

[54] SciPy, "Scientific Tools for Python," http://www.scipy.org/, 2014.

[55] VideoLan Organization, "x264 Home Page," http://www.videolan.org/developers/x264.html, 2014.

[56] S. Baudry, P. Nguyen, and H. Maitre, "Channel coding in video watermarking: use of soft decoding to improve the watermark retrieval," in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 3. IEEE, 2000, pp. 25–28.

[57] R. Parviainen and P. Parnes, "Large scale distributed watermarking of multicast media through encryption," in *Communications and Multimedia Security Issues of the New Century*. Springer, 2001, pp. 149–158.