MULTIMEDIA ENCRYPTION AND WATERMARKING

Borko Furht
Edin Muharemagic
Daniel Socek

Springer

MULTIMEDIA ENCRYPTION AND WATERMARKING

MULTIMEDIA SYSTEMS AND APPLICATIONS SERIES

Consulting Editor

Borko Furht
Florida Atlantic University
borko@cse.fau.edu

Recently Published Titles:

ADVANCED WIRED AND WIRELESS NETWORKS edited by Tadeusz A. Wysocki, Arek Dadej and Beata J. Wysocki; ISBN: 0-387-22847-0

CONTENT-BASED VIDEO RETRIEVAL: ***A Database Perspective*** by Milan Petkovic and Willem Jonker; ISBN: 1-4020-7617-7

MASTERING E-BUSINESS INFRASTRUCTURE, edited by Veljko Milutinović, Frédéric Patricelli; ISBN: 1-4020-7413-1

SHAPE ANALYSIS AND RETRIEVAL OF MULTIMEDIA OBJECTS by Maytham H. Safar and Cyrus Shahabi; ISBN: 1-4020-7252-X

MULTIMEDIA MINING: ***A Highway to Intelligent Multimedia Documents*** edited by Chabane Djeraba; ISBN: 1-4020-7247-3

CONTENT-BASED IMAGE AND VIDEO RETRIEVAL by Oge Marques and Borko Furht; ISBN: 1-4020-7004-7

ELECTRONIC BUSINESS AND EDUCATION: ***Recent Advances in Internet Infrastructures***, edited by Wendy Chin, Frédéric Patricelli, Veljko Milutinović; ISBN: 0-7923-7508-4

INFRASTRUCTURE FOR ELECTRONIC BUSINESS ON THE INTERNET by Veljko Milutinović; ISBN: 0-7923-7384-7

DELIVERING MPEG-4 BASED AUDIO-VISUAL SERVICES by Hari Kalva; ISBN: 0-7923-7255-7

CODING AND MODULATION FOR DIGITAL TELEVISION by Gordon Drury, Garegin Markarian, Keith Pickavance; ISBN: 0-7923-7969-1

CELLULAR AUTOMATA TRANSFORMS: ***Theory and Applications in Multimedia Compression, Encryption, and Modeling***, by Olu Lafe; ISBN: 0-7923-7857-1

COMPUTED SYNCHRONIZATION FOR MULTIMEDIA APPLICATIONS, by Charles B. Owen and Fillia Makedon; ISBN: 0-7923-8565-9

STILL IMAGE COMPRESSION ON PARALLEL COMPUTER ARCHITECTURES by Savitri Bevinakoppa; ISBN: 0-7923-8322-2

INTERACTIVE VIDEO-ON-DEMAND SYSTEMS: ***Resource Management and Scheduling Strategies***, by T. P. Jimmy To and Babak Hamidzadeh; ISBN: 0-7923-8320-6

MULTIMEDIA TECHNOLOGIES AND APPLICATIONS FOR THE 21st CENTURY: ***Visions of World Experts***, by Borko Furht; ISBN: 0-7923-8074-6

MULTIMEDIA ENCRYPTION AND WATERMARKING

by

Borko Furht
Edin Muharemagic
Daniel Socek
Florida Atlantic University
Boca Raton, FL, USA

Springer

Borko Furht / Edin Muharemagic / Daniel Socek
777 Glades Road
Florida Atlantic University
Dept. of Computer Science & Engineering
P.O.Box 3091
Boca Raton FL 33431

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available
from the Library of Congress.

MULTIMEDIA ENCRYPTION AND WATERMARKING
by Borko Furht, Edin Muharemagic, Daniel Socek

Multimedia Systems and Applications Series Volume 28

ISBN-10: 0-387-24425-5 e-ISBN-10: 0-387-26090-0
ISBN-13: 978-0-387-24425-9 e-ISBN-13: 978-0-387-26090-7

Printed on acid-free paper.

© 2005 Springer Science+Business Media, Inc.
All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, Inc., 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now know or hereafter developed is forbidden.
The use in this publication of trade names, trademarks, service marks and similar terms, even if the are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed in the United States of America.

9 8 7 6 5 4 3 2 1 SPIN 11053972, 11431817

springeronline.com

To women of our lives.

Drasan, Janamir, and Disan
(encrypted using a naïve approach)

Sandra, Mirjana, and Sandi
(decrypted)

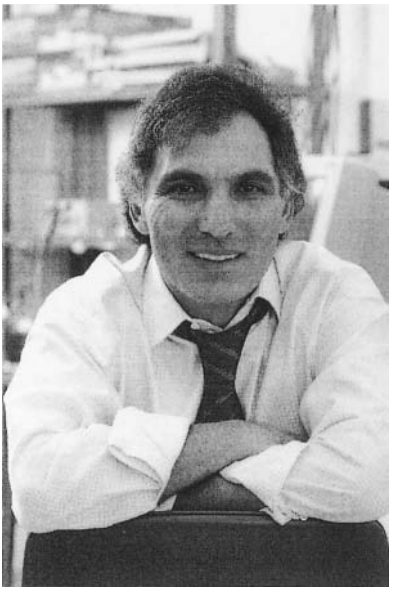
Preface

The purpose of this book is to present a comprehensive survey of contemporary multimedia encryption and watermarking techniques, which enable a secure exchange of multimedia intellectual property. The book is intended both for researchers and practitioners in the field, and for scientists and engineers involved in designing and developing systems for the protection of digital multimedia content. The book can also be used as the textbook for graduate courses in the area of multimedia security.

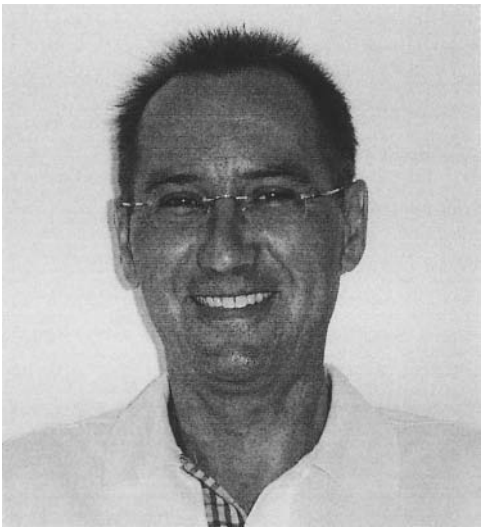
The book is comprised of 14 chapters divided into 3 parts. Part I, Digital Rights Management (DRM) for Multimedia, introduces DRM concepts and models for multimedia content protection. It also presents key players in multimedia content protection. Part II, Multimedia Cryptography, presents an overview of modern cryptography, and focuses on modern image, video, speech, and audio encryption techniques. It also presents an advanced concept of visual and audio sharing and related techniques. Part III, Digital Watermarking, introduces the concept of watermarking for multimedia, classifies watermarking applications, and evaluates various multimedia watermarking concepts and techniques. In this part, digital watermarking techniques for binary images are described as well.

Borko Furht, Edin Muharemagic, and Daniel Socek

Authors

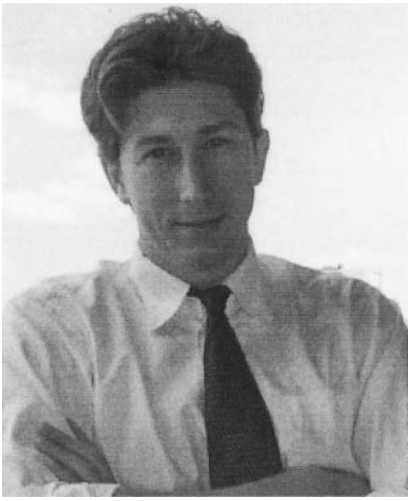


Borko Furht is a professor and chairman of the Department of Computer Science and Engineering at Florida Atlantic University (FAU) in Boca Raton, Florida. Before joining FAU, he was a vice president of research and a senior director of development at Modcomp (Ft. Lauderdale), a computer company of Daimler Benz, Germany, and a professor at University of Miami in Coral Gables, Florida. Professor Furht received Ph.D. degrees in electrical and computer engineering from the University of Belgrade. His current research is in multimedia systems, wireless multimedia, video coding and compression, and video databases. He is the author of numerous books and articles in the areas of multimedia, computer architecture, real-time computing, and operating systems. He is a founder and editor-in-chief of *the Journal of Multimedia Tools and Applications* (Kluwer Academic Publishers). He is currently one of principal investigators of two multi-year projects – Federal Earmark project on Technologies for Coastline Security, and the project on One Pass to Production, funded by Motorola. He has received several technical and publishing awards, and has consulted for many high-tech companies including IBM, Hewlett-Packard, Xerox, General Electric, JPL, NASA, Honeywell, and RCA. He has also served as a consultant to various colleges and universities. He has given many invited talks, keynote lectures, seminars, and tutorials.



Edin Muharemagic is an independent consultant and an adjunct professor at Florida Atlantic University. He has over fifteen years of industrial experience in software design, development, technical management, and teaching. He currently works in IBM on the embedded voice recognition technologies. Previously, he held senior technical and management positions at IBM and RadiSys Corporation.

Edin Muharemagic received the B.S degree in mathematics from the University of Belgrade, Yugoslavia, the M.S. degree in computer science from New Jersey Institute of Technology, Newark, New Jersey, and Ph.D. degree in computer engineering from Florida Atlantic University, Boca Raton, Florida. His research interests include multimedia security, information hiding and digital watermarking. He has published several book chapters, and journal and conference papers.



Daniel Socek is a research assistant and a Ph.D. candidate in the Department of Computer Science and Engineering at Florida Atlantic University (FAU). He is also affiliated with the Center for Cryptology and Information Security at FAU. He graduated from University of Nebraska–Lincoln in 2000 with the B.Sc. degree in computer science, and received the M.Sc. degree in mathematics from Florida Atlantic University in 2002. His current research interests include cryptography, multimedia security, audiovisual secret sharing, lattice reduction-based cryptanalysis, and marine surveillance security. For his outstanding academic and research performance, he was awarded the Graduate Fellowship for Academic Excellence and the Dr. Daniel B. and Aural B. Newell Doctoral Fellowship in 2003 and 2004, respectively. He worked and served as a consultant for various IT companies such as IBM, Panasonic, Matsushita Electric Industrial, and Avaton/Crypton. He has given various talks and tutorials at conferences, workshops and seminars.

Contents

Part I

Introduction to Multimedia Security

Chapter 1

DIGITAL RIGHTS MANAGEMENT FOR MULTIMEDIA

1.1 INTRODUCTION TO DIGITAL RIGHTS MANAGEMENT

Digital rights management (DRM) is a collection of techniques and technologies that enable technically enforced licensing of digital information. Specifically, DRM for multimedia enables the secure exchange of multimedia intellectual property, which includes copyright-protected music, video, or text in digital form over the Internet or other electronic media, such as DVDs, CDs, removable disks, and mobile networks.

The main players in a DRM system are multimedia content creators, the content owners, and the consumers (or clients). The creators and the owners may not be the same. The multimedia content needs to be transferred from the owner to the consumer. Consequently, a DRM solution is based on a DRM architecture (sometimes referred as DRM reference architecture [7] or Content Protection System Architecture [4]). The components of a DRM architecture are typically divided between the server at the content provider, the server at the DRM service provider, and the hardware at the consumer site. Three major components of a typical DRM architecture for multimedia content include the content server, the license server, and the client, as illustrated in Figure 1.1 [13].

The Content Server includes the content database, the content information database, and the DRM or content packager. The content database is a collection of multimedia content, such as movies, audio and music files, multimedia files, etc. These files may be already stored in the correct format that is ready for distribution through the DRM system (for example these files are already encrypted), or these files can be converted on demand in the correct form.

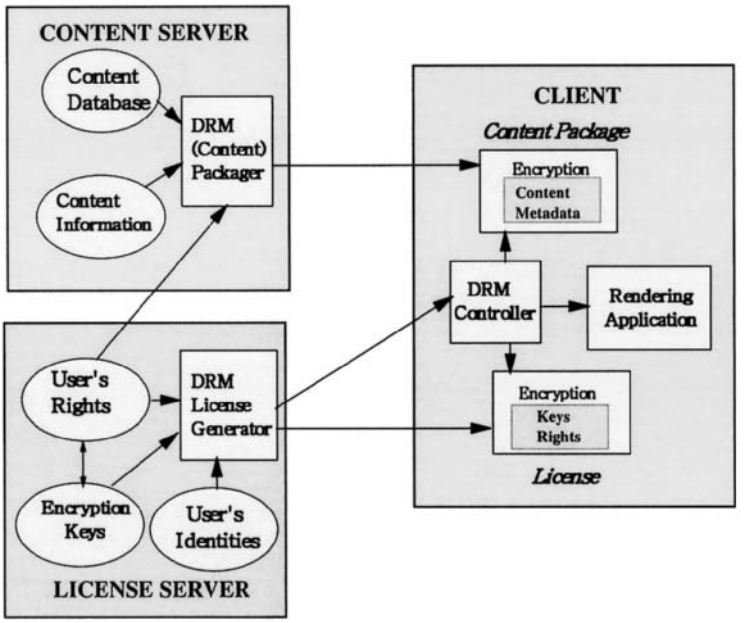


Figure 1.1. A typical DRM architecture for multimedia (adapted from [13]).

The Content Information Database is a collection of metadata, which describe the content

The DRM or the Content Packager prepares the content for the DRM distribution through the system. It typically adds encryption or watermarking information into the content before the content repository to the database, or before the content distribution.

The License Server contains the information about consumers (or users) who need to access and receive the multimedia content from the content database. The basic information consists of the user's identification and the user's rights to access the content database. A typical implementation of the License Server is based on a set of encryption keys, which are utilized for user authentication and for the decryption of the multimedia content. Based on the user identity and encryption keys, the License Generator creates licenses and sends them to the user, as illustrated in Figure 1.1.

At the client site, the main component is the DRM Controller, which performs the following operations: (a) receives the user request to access the content, (b) gets the user's identity information and receives the license from the License Server, and (c) retrieves the encryption keys from the license and decrypts the content.

The core technologies, which are used to implement DRM systems for multimedia include encryption and watermarking. The main focus of this book is to describe and evaluate modern trends and techniques in the areas of audio, image, and video encryption and watermarking. Multimedia encryption techniques are described in Part II of the book, while multimedia watermarking techniques are presented in Part III. Brief introduction of multimedia encryption and watermarking, and their relationship is given below.

A multimedia encryption technique encrypts the multimedia content (such as audio, video, image, etc.) with the goal to prevent an unauthorized user to access the content.

A multimedia watermarking technique creates a metadata, which contains the information about the multimedia content to be protected, and then hides this metadata within that content. Figure 1.2 illustrates a main difference between encryption and watermarking for a video content.

In the case of video encryption, the DRM Packager will first encrypt the video content, and the DRM Controller through its decryption application will decrypt the video content, which will be played out on a video player. In the case of watermarking, the DRM Packager will insert a metadata as a watermark into the video content, while the DRM Controller through its watermark extraction application will extract this metadata (or watermark) from the video content.

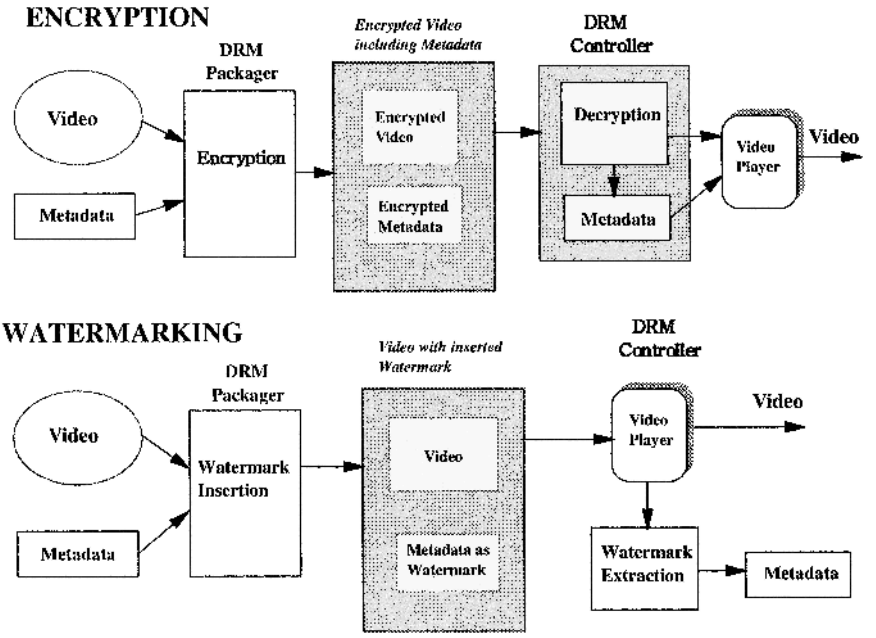


Figure 1.2. The difference between multimedia encryption and watermarking (adapted from [13]).

Modern DRM systems for multimedia combine both encryption and watermarking. We describe two possible scenarios how to do it [13]. One possible way is to encrypt the watermark itself. This is beneficial for applications where the watermark contains data about the content ownership, which must be protected against alteration.

The other scenario of combining watermarking with encryption is to first insert a watermark into the multimedia content and then to encrypt it, as shown in Figure 1.3.

In summary, the DRM for multimedia has three main objectives, which can be achieved using encryption and watermarking technologies [16]:

- Authentication to assure the credibility of multimedia information

- Confidentiality to secure privacy of the content transmission, and
- Copy control to protect multimedia data from illegal distribution,

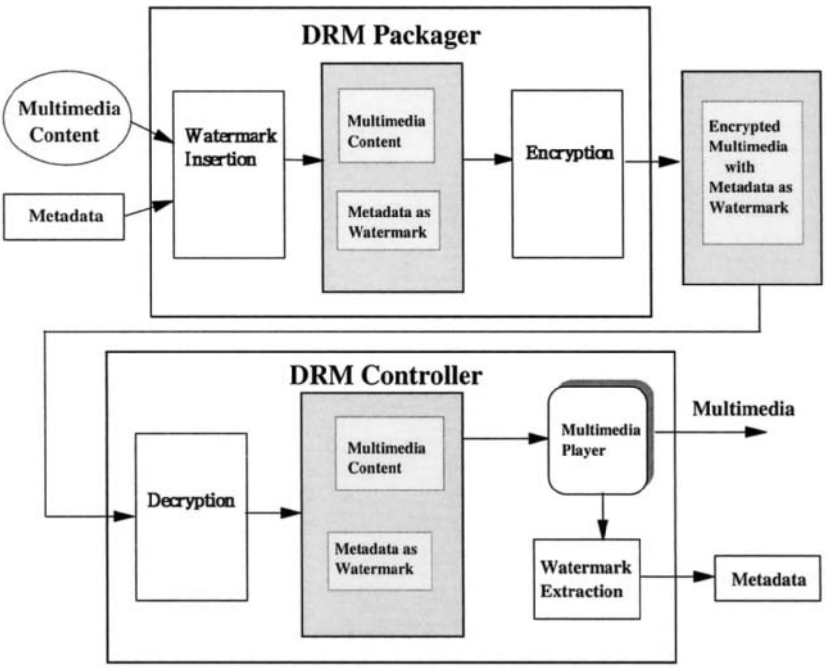


Figure 1.3. Combining multimedia watermarking with encryption. (adapted from [13]).

1.2 MODELS FOR CONTENT PROTECTION

There are two basic models for content protection: point-to-point and end-to-end content protection systems [9]. Most current content protection systems and standards are based on point-to-point model. Content is always encrypted when it is stored on a device, or when transmitted between devices. Each device decrypts the content that it received at the input, decompresses the content, and then again re-encrypts the content for the next component in the system. Figure 1.4 presents a point-to-point system with three components.

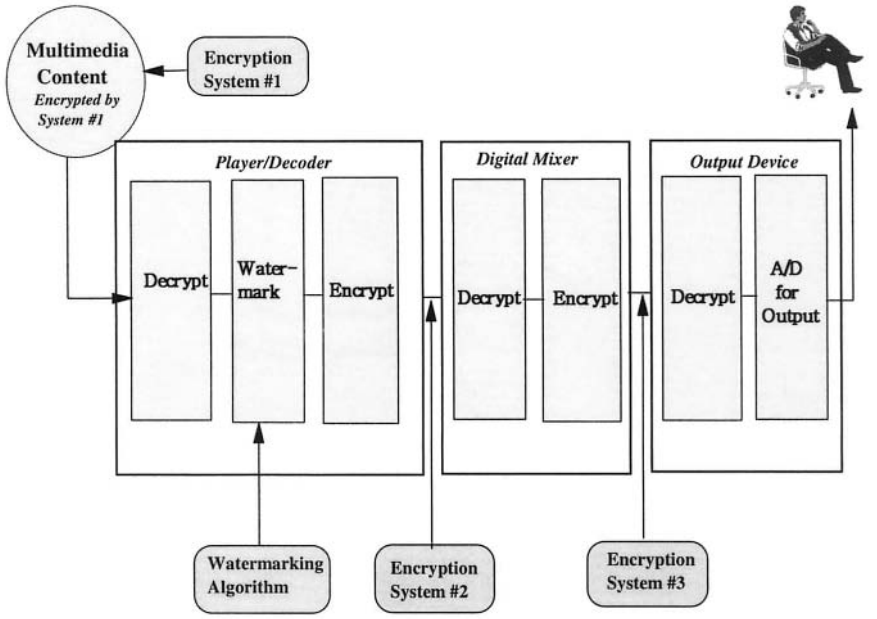


Figure 1.4. Point-to-point content protection system (adapted from [9]).

Point-to-point content protection systems provide security, only if all supported devices and protocols are secure. If the keys from one device's input are compromised, even if content owners are aware of the attack, other devices will continue to output content using these keys.

End-to-end content protection systems also use devices which are encrypted individually, like in point-to-point systems; but in addition they implement end-to-end validation. In these systems, the initial decoder device validates how the content will be used as it is transmitted through the system [9]. One possible end-to-end validation can be implemented in such a way that the player/decoder provide an interface through which the content's security code can identify and query downstream objects, as illustrated in Figure 1.5.

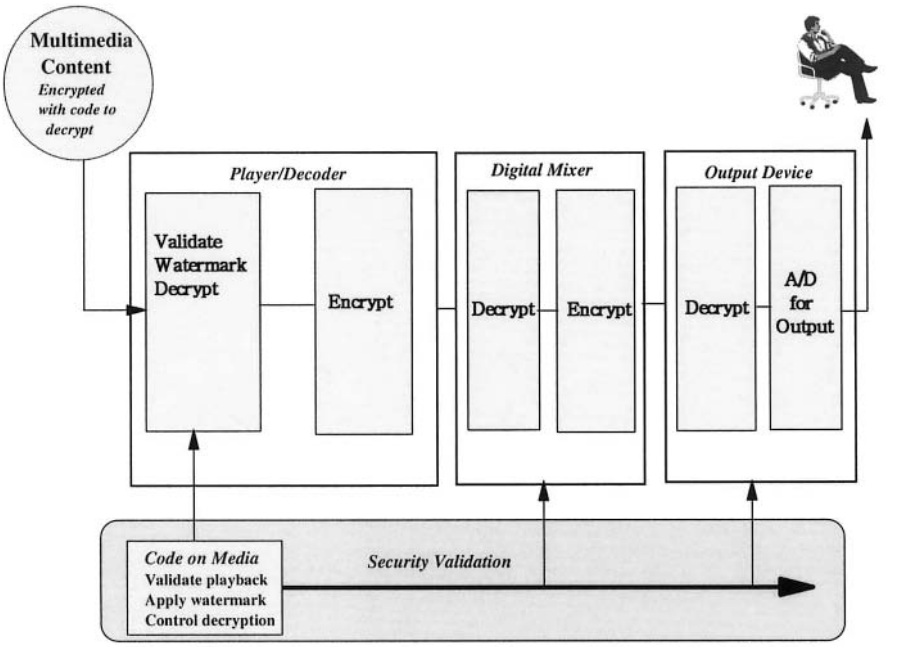


Figure 1.5. End-to-end content protection system (Adapted from [9]).

In Figure 1.5, compromises prior to content delivery to the user can be handled using the content-controlled programmable risk-management approaches [9].

1.3 CONTENT PROTECTION SOLUTIONS FOR MULTIMEDIA

In this section we briefly present an overview of technical solutions for audio and video protection, which are based on models described in 1.2. The selected solutions are listed in Table 1.1.

Table 1.1. Content Protection Solutions for A/V (adapted from [4])

SOLUTION	WHAT IS PROTECTED	BRIEF DESCRIPTION
CSS	Video on DVD-ROM	CSS protects video, which is decrypted during playback on the DVD player
CPRM	Video or audio on a number of media types, including DVD-R/RW/RAM	Audio/Video content is re-encrypted before recording on a DVD recordable disc. During playback, the player derives the decryption key.
HDCP	Digital Visual Interface (DVI)	Video transmitter authenticates the receiver and establishes shared secrets with it. Audio/video content is encrypted across the interface. The encryption key is renewed frequently.
SPDC	High-definition video on optical disks	SPDC allows each media disc to carry its own decoding software. This content code can interact with users, query the player, and control the playback process.

The Content Scramble System (CSS) is typically used for protection of DVD video. It is designed by Matsushita to encrypt MPEG-2 video. CSS is implemented in the player based on a sample, fixed security policy for all content. This policy allows any device with valid keys to decrypt all media valid in its region. The architecture of a content player using CSS is shown in Figure 1.6.

In the CSS system, the content is first compressed, then encrypted, and distributed on read-only media. The player is preloaded with a pair of keys – one key is unique to the device, and the other is unique to the MPEG file being decrypted. After decryption, the content is sent to the output interface. The output interface can be unprotected, or its protection is independent of the protection used on the media.

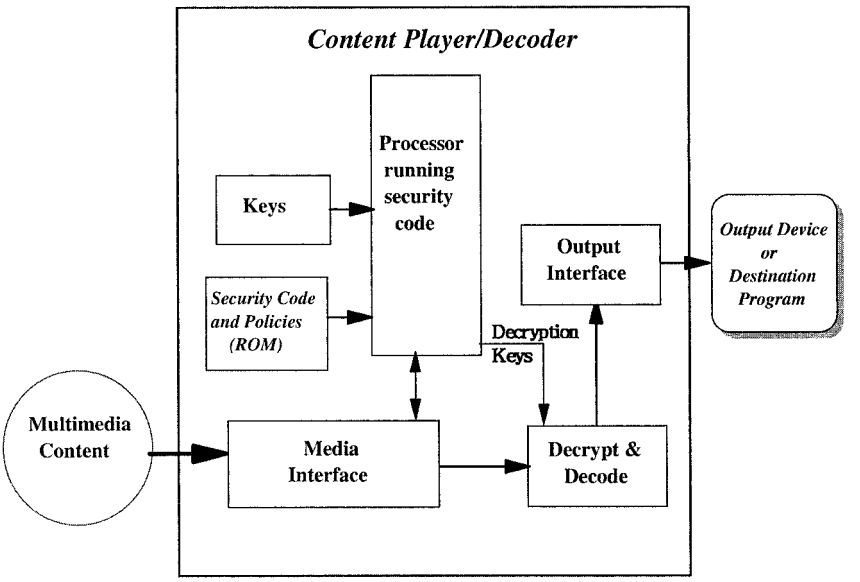


Figure 1.6. Architecture of a content player using the CSS solution.

In the CSS system, each manufacturer is assigned a distinct master key. If this key is compromised, it must be replaced by another key; however the entire installed base of DVD players becomes obsolete.

The CPRM (Content Protection for Recordable Media) system is shown in Figure 1.7 [Content Protection]. The system consists of a Service Provider, a Security Provider, and a Client. The service provider delivers content prepared for recording on CPRM media. The content is encrypted with Cryptomeria Cipher (C2) using a randomly selected title key. In order to play the content on CPRM devices, there is an additional operation – media binding. During this operation, the Security Provider receives the Media ID and the Media Key Block for specific piece of CPRM media from the client. Based on this information, the Security Provider calculates the Media Unique Key, which then encrypts the Title Key. It sends the encrypted Title Key to the Client, and the Client places the encrypted title key along with the encrypted content from the Service Provider

on the CPRM media. At this point, the content can be played by any CPRM-compliant playing device.

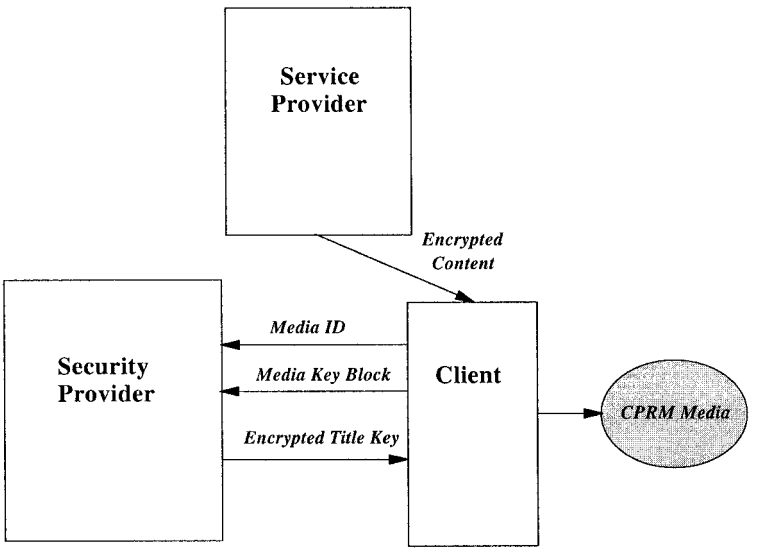


Figure 1.7. An example of the system using CPRM.

High-bandwidth Digital Content Protection (HDCP) is a specification developed by Intel Corporation to protect digital entertainment content across the DVI/HDMI interface. The HDCP specification provides a robust, cost-effective and transparent method for transmitting and receiving digital entertainment content to DVI/HDMI-compliant digital displays [High-bandwidth]. The HDCP license contains robustness and compliance rules that ensure that HDCP implementations both protect the confidentiality of keys and other values from compromise as well as delivers the desired protection for high-value video content.

The Self-Protecting Digital Content (SPDC) architecture protects high-definition video distributed on next-generation optical discs [9]. The player provides a simple, specialized virtual machine, but neither understands nor controls the decoding process. If a particular disc or device is compromised, subsequent content can carry a new security system that addresses this vulnerability. This

approach is referred as "programmable security", which allows publishers to add new countermeasures and improve security after a standard has been widely adopted. Players include a simple virtual machine with APIs that provide data about the playback environment, such as player information, software versions, types of output device, and user commands.

An overview of other protection system architectures including satellite, cable, terrestrial, and Internet protection systems is given in [2].

1.4 KEY PLAYERS IN MULTIMEDIA CONTENT PROTECTION

As a consequence of the war between the content industry and hackers, the copyright industry has significantly grown. According to the survey reported in [2], the estimated US losses due to copyright piracy were about $10 trillions in 2002.

The multimedia security problem involves interdisciplinary collaboration among researchers in information technology and entertainment and consumer electronics industries. Many forums and working groups have been formed, which are dealing with standards in digital content protection and other significant issues in this field. Table 1.2, adapted from [3][16] presents the key players and their main activities.

Table 1.2. Key Players in Digital Content Protection (adapted from [2][16])

FORUM or GROUP	MAJOR FOCUS	Web Site
Copyright Protection Technical Working Group (CPTWG)	Focus on standardizing copy protection tools for digital interfaces and watermarking for DVD video and audio content.	cptwg.org
Digital Audio-Visual Council	DAVIC standard contains a general cop-protection framework baseline document for digital TV and interactive applications.	www.davic.org
Secure Digital Music Initiative's	Open technology specifications to protect legitimate playing, storing, and distribution of digital music.	www.sdmi.org
Open Platform Initiative for Multimedia Access	Standardizes an open generic framework for access control and content management and protection tools for Internet and pay TV applications	www.cselt.it
DVD Forum	The Forum's purpose is to exchange and disseminate ideas and information about the DVD format and its technical capabilities, improvements and innovations.	www.dvdforum.org
Digital Video Broadcasting Project	DVB is an industry-led consortium of over 260 broadcasters, manufacturers, network operators, software developers, regulatory bodies and others in over 35 countries committed to designing global standards for the global delivery of digital television and data services.	www.dvb.org

REFERENCES

[1] "Content Protection for Recordable Media Specification," *Intel, IBM, Matsushita, Toshiba*, August 5, 2004.

[2] A.M. Eskicioglu, "Protecting Intellectual Property in Digital Multimedia Networks," *IEEE Computer*, July 2003, pp. 39-45.

[3] A.M. Eskicioglu and E.J. Delp, "Protection of Multimedia Content in Distribution Networks," Chapter 1 *in "Multimedia Security Handbook," by B. Furht and D. Kirovski*, CRC Press, Boca Raton, Florida, 2005.

[4] A.M. Eskicioglu and E.J. Delp, "An Overview of Multimedia Content Protection in Consumer Electronics Devices," *Signal Processing: Image Communication*, Vol. 16, 2001, pp. 681-699.

[5] D. Emanuel and M.S. Kankanhalli, "Digital Rights Management Issues for Video, Chapter 26 *in "Multimedia Security Handbook," by B. Furht and D. Kirovski*, CRC Press, Boca Raton, Florida, 2005.

[6] B. Furht and D. Kirovski, *"Multimedia Security Handbook,"* CRC Press, Boca Raton, Florida, 2005.

[7] F. Hartung and F. Ramme, "Digital Rights Management and Watermarking of Multimedia Content for M-Commerce Applications," *IEEE Communications Magazine*, November 2000, pp. 78-84.

[8] "High-bandwidth Digital Content Protection System," *Digital Content Protection LLC*, June 2003, www.digital-cp.com.

[9] P. Kocher, J. Jaffee, B. Jun, C. Laren, and N. Lawson, "Self-Protecting Digital Content," *White Paper, Cryptography Research*, 2004.

[10] R.H. Koenen, J. Lacy, M. Mackay, and S. Mitchell, "The Long March to Interoperable Digital Rights Management," *Proceedings of the IEEE*, Vol. 92, No. 6, June 2004, pp. 883-897.

[11] D. Kundur, C-Y. Lin, B. Macq, and H. Yu, "Special Issue on Enabling Security Technologies for Digital Rights Management, " *Proceedings of the IEEE*, Vol. 92, No. 6, June 2004.

[12] J. Lotspech, "Digital Right Management for Consumer Devices," Chapter 23 *in "Multimedia Security Handbook," by B. Furht and D. Kirovski*, CRC Press, Boca Raton, Florida, 2005.

[13] B. Rosenblatt, B. Trippe, and S. Monney, *"Digital Rights Management: Business and Technology,"* M&T Books, New York, 2002.

[14] C.B.S. Traw, "Technical Challenges of Protecting Digital Entertainment Content," *IEEE Computer Magazine*, July 2003, pp. 72-78.

[15] M. Trimeche and F. Chebil, "Digital Right Management for Visual Content in Mobile Applications," *IEEE*, 2004, pp 95-98.

[16] H. H. Yu, D. Kundur, and C-Y. Lin, Spies, Thieves, and Lies: The Battle for Multimedia in the Digital Era," *IEEE Multimedia*, July-September 2001, pp. 8-12.

Part II

Multimedia Cryptography

Chapter 2

INTRODUCTION TO MULTIMEDIA ENCRYPTION

2.1 MULTIMEDIA AND CRYPTOGRAPHY

In the recent years, there has been a tremendous improvement and emergence of technologies for communications, coding, and retrieval of digital multimedia. Such an environment has allowed for the realization of many fascinating multimedia applications related to nearly all aspects of life. Almost instantaneous delivery of entertainment videos, pictures and music is available to everyone who is connected to a multimedia distribution system. Businesses and other organizations are now able to perform real-time audioconferencing and videoconferencing, even over a non-dedicated channel. By connecting to a medical imaging database system, the experts, even the ones from remote areas, can instantaneously receive and review relevant medical images using the power of image coding and image retrieval techniques. However, many multimedia distribution networks are open public channels and, as such, are highly insecure. An eavesdropper can conveniently intercept and capture the sensitive and valuable multimedia content traveling in a public channel. Fortunately, the magic art of cryptography can help prevent this.

In general, multimedia security is achieved by a method or a set of methods used to protect the multimedia content against unauthorized access or against unauthorized distribution. These methods are heavily based on cryptography and they provide either communication security, or security against piracy (DRM and watermarking). Communication security of multimedia data can be accomplished by means of conventional cryptography. Multimedia data could be entirely encrypted using a cryptosystem such as DES [9], IDEA [10] or AES [11]. In many cases, when the multimedia is textual or static data, and not a real-time streaming media, we can treat it as an ordinary binary data and use the conventional

encryption techniques. Encrypting the entire multimedia stream using standard encryption methods is often referred to as the *naïve approach.* The naïve approach is usually suitable for text, and sometimes for small bitrate audio, image, and video files that are being sent over a fast dedicated channel. Secure Real-time Transport Protocol [1], or shortly SRTP, is an application of the naïve approach. In SRTP, multimedia data is packetized and each packet is individually encrypted using AES. The naïve approach enables the same level of security as that of the utilized conventional cryptosystem. Unfortunately, encrypting the entire bitstream is typically not possible for higher bitrate multimedia, especially when the transmission is done over a non-dedicated channel.

Due to a variety of constraints, communication security for streaming multimedia is harder to accomplish [6,7,8]. Such constraints include real-time processing, non-dedicated channels with limited or varying bandwidth, high multimedia bitrate, and more. Thus, a communication encryption of many video and audio multimedia is not simply the application of established conventional encryption algorithms to their binary sequence. It involves careful analysis to determine and identify the optimal encryption method. Current research is focused towards exploiting the format specific properties of many standard multimedia formats in order to achieve the desired performance. This is referred to as the *selective encryption.* This type of encryption is obviously preferred when compression and decompression algorithms can hardly keep up with the required bitrate, even when these algorithms are accelerated by a dedicated hardware. In some cases, encryption and decryption algorithms could also be accelerated by hardware. However, software implementations are often preferred due to their flexibility and low cost. Figure 2.1 shows the logical stages that are taken during both the naïve approach and the selective approach.

The naïve approach is obviously more straightforward to implement, but the entire bitstream is passing through the computationally expensive encryption and decryption stages. Although more complex to implement, selective encryption will only process the selected bits through encryption and decryption stages. More recently, full encryption approaches based on chaotic maps have been proposed for securing the multimedia.

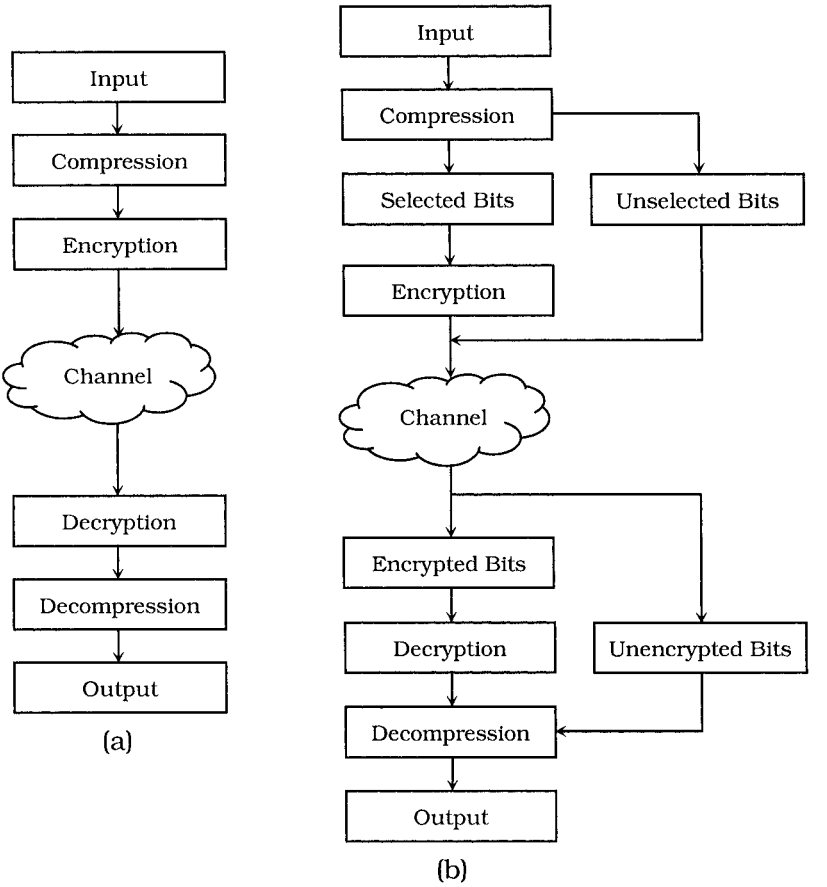


Figure 2.1. Stages taken during multimedia encryption using (a) the naïve approach, and (b) the selective approach.

Chaos-based cryptography had been studied for quite some time, but the encryption approaches using 1-D, 2-D and even 3-D chaotic maps specifically designed for encrypting digital images and videos had been only recently proposed. Chaos-based multimedia encryption algorithms are of high interest due to their extremely fast performances and suitability for multimedia data.

We have already mentioned several reasons as to why we need multimedia-specific cryptosystems. Most of the traditional cryptosystems were designed specifically to secure and encrypt text messages. Digital multimedia data, however, have much different

properties, which make them less suitable for many conventional cryptosystems. Multimedia data includes images, videos, speech, and audio, which are usually very large-sized and bulky. When encrypting such bulky data, the performance overhead that is introduced with some of the conventional ciphers is generally too expensive for real-time applications. Secondly, an extremely high data redundancy is often present in many uncompressed images and videos. Consequently, some of the conventional block ciphers may fail to obscure all visible information unless advanced modes of operation are used. Figure 2.2 shows how AES that operates in its basic ECB mode fails to disguise the image. As one can see, if the more complex modes are used, AES is more effective. However, modes that are more computationally complex consequently run slower. Different modes of block ciphers will be discussed further in the next chapter.

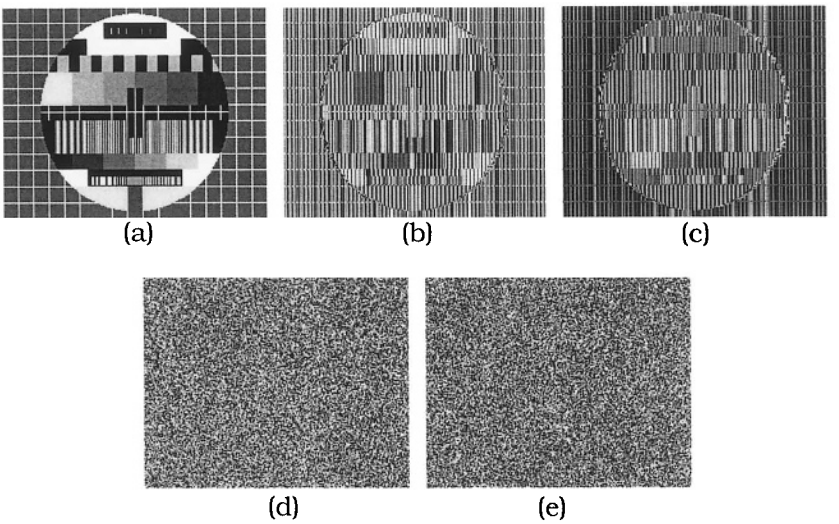


Figure 2.2. Encrypting a redundancy rich uncompressed image: (a) the original image, (b) the image encrypted with AES-128 in ECB mode, (c) the image encrypted with AES-256 in ECB mode, (d) the image encrypted with AES-128 in CBC mode, and (e) the image encrypted with AES-256 in CBC mode.

In the case of digital videos, consecutive frames are extremely similar and most likely only few pixels (if any) would differ from frame to frame. The conventional ciphers are designed with a good avalanche

property, however, in the case of videos, an extremely strong avalanche property is required. Furthermore, chosen/known-plaintext attack can be easily performed for videos, and a cipher that is extremely robust to chosen/known-plaintext attacks is desired. Another drawback of conventional cryptosystems relates to the compressed multimedia. Multimedia compression and encryption are usually very incompatible. Encrypting the multimedia content before compression removes a lot of redundancy and this results in a very poor compression ratio. On the other hand, encrypting the data after compression destroys the codec format, which causes decoders to crash. Finally, for many applications, we would like to have very light encryption that preserves some perceptual information. This is impossible to achieve with conventional cryptosystems alone, which most likely degrade the data to a perceptually unrecognizable content. Having cryptosystems that preserve a rough view of the high-quality media material is preferred for many real-life applications, such as video pay-per-view system in which a degraded but visible content could potentially influence a consumer to order certain paid services.

2.2 SECURITY VS. PERFORMANCE

Undoubtly, implementing security based on a cryptographic protocol introduces a performance overhead during the multimedia processing. The size of an overhead depends on many factors. In most cases, utilizing only the general software or compiler based optimization techniques yields only modest improvements. The complexity of an encryption/decryption algorithm is one of the major factors affecting the secure multimedia system performance. Clearly, a fast, yet secure algorithm is desired. In fact, the main goal of multimedia encryption is to significantly reduce the performance overhead while maintaining the desired level of security. Only then can one expect the secure real-time delivery of high-quality and large bitrate multimedia over a non-dedicated channel. Unfortunately, this is not easy to accomplish.

Researchers and experts put a lot of effort in designing good multimedia encryption algorithms. As a result, there is a significant number of proposals that rely on the selective encryption. Once a selection of bits to be encrypted is made, a conventional encryption could be used to secure them. By a "conventional cryptosystem" we mean a cryptosystem that is either one of the encryption standards

(DES or AES), or a well-established cryptosystem that was designed by the cryptography experts and that has been around for a while. Many cryptographic systems fulfill the aforementioned credibility requirements. Unfortunately, there are far more of them that do not meet these requirements. It is often dangerous to blindly trust a cryptosystem that had been recently proposed, as the cryptographic community needs some time to thoroughly investigate its security and possibilities of an attack. In this book, we will describe few conventional cryptosystems that can be safely used as a basis for many multimedia selective encryption approaches. These cryptosystems withstood many years of attacks and we can safely assume that their security is likely to remain strong. Modifying or simplifying conventional cryptosystems in order to improve the multimedia encryption performance is usually a bad idea. It is as bad an idea as if a construction worker would suddenly decide to build a wall using the cardboard boxes instead of bricks to save the cost or to speed up the construction process. Before you know it, someone would almost effortlessly break the wall, just as someone would almost effortlessly break the "simplified" cryptosystem. Yet, a number of early multimedia security solutions used far oversimplified encryption algorithms in order to produce faster performances. This kind of cryptography does not provide security. It hardly provides a temporary inconvenience for the attacker.

This book, however, is not about cryptography. Cryptography is complex, difficult, and a highly mathematically involved discipline. This book is about applying cryptography to achieve the desired multimedia security and protection. While some major concepts of cryptography will be discussed to provide a better understanding of topics ahead, the primary focus of this part of the book is on the multimedia encryption techniques, their performance, and their security.

Although slightly outdated, "Handbook of Applied Cryptography" by van Oorschot, Menezes, and Vanstone [2] is still a top technical reference to the subject of general cryptography, while Bruce Schneier's "Applied Cryptography" [3] is still an excellent resource that in addition includes the C code for many important cryptography-related algorithms. An excellent new book on the subject is also "Fundamentals of Computer Security" by Pieprzyk, Hardjono, and Seberry [4]. Finally, Doug Stinson's "Cryptography: Theory and Practice" [5] is one of the best textbooks about

cryptography that covers the core material and includes numerous examples and exercises.

2.3 LEVELS OF SECURITY

A given cryptosystem provides a certain level of security. Most available cryptographic systems are fully or partially scalable, in the sense that one can choose different security levels. Scalability is usually achieved by allowing variable key sizes or by allowing different number of iterations, or rounds. A higher level of security is achieved with larger key sizes or larger number of rounds. Consequently, an algorithm becomes more complex and slower. Therefore, it is important to carefully assess the value and importance of the multimedia content that needs to be encrypted. According to the content value, the cryptosystem can be properly scaled to allow an optimal performance-security ratio.

There is yet another important consideration in the case of multimedia encryption. Applying selective encryption has its own scalability chart. Obviously, selecting all bits for encryption is inefficient, but certainly the most secure choice. The amount of selected bits and type of selected bits uniquely determine the complexity of an attack. This type of scalability is much harder to grade, and the tools for its assessment are not yet fully developed nor established. The reason for this is that the conventional, non-multimedia encryption algorithms have a long history. Analytical methods and tools were developed over a course of many years, and in many instances rigorous mathematical proofs allowed for proper evaluation. On the other hand, multimedia specific encryption algorithms are relatively new. First such algorithms started to appear in the mid-1990s, and in most cases, solid methods for rigorous security analysis of such algorithms are simply not available. An additional cause for this is the high complexity of many encoded multimedia formats. For example, some of the available video codecs produce such intricate data that their true mutual correlation is extremely complex to understand. To evaluate how much the non-encrypted data can reveal about the encrypted bits is consequently a hard problem.

The problem of rigorous security analysis of the proposed multimedia encryption algorithms is interdisciplinary in nature. It requires proficiency in multimedia processing, as well as a

proficiency in cryptography and cryptanalysis. In this book we try to overcome these problems as much as possible, by providing careful analysis of the proposed image, video, audio, and speech encryption algorithms. To our knowledge, that is what makes this book unique.

2.4 DEGRADATION AND SCRAMBLING

Deciding upon what level of security is needed is harder than it looks. To identify an optimal security level, we have to carefully compare the cost of the multimedia information to be protected versus the cost of the protection itself. If the multimedia to be protected is not that valuable in the first place, it is sufficient to choose a relatively light level of encryption. On the other hand, if the multimedia content is highly valuable or represents industrial, governmental or military secrets, the cryptographic security level must be the highest possible.

For many real-world applications (such as pay-per-view) the content data rate is typically very high, but the monetary value of the content may not be high at all. Thus, very expensive attacks are not attractive to adversaries, and lightweight encryption, often called *degradation*, may be sufficient for distributing such multimedia content. For such applications, DRM is of much more interest since the content owner and the content provider both seek to enforce the copyrights and the distribution rights. When degradation is applied to a multimedia content, the content is usually still perceptible to some degree. For instance, one may still see the objects in a degraded video, but the visual quality should be unacceptable for entertainment purposes (see Figure 2.3-b). In contrast, applications such as videoconferencing or videophone (or even Internet phone) may require a much higher level of confidentiality. If the videoconference is discussing important industrial, governmental or military secrets, then the cryptographic strength must be substantial. However, maintaining such a high level of security and still keeping the real-time and limited-bandwidth constraints is not easy to accomplish.

Another form of lightweight encryption is so-called *scrambling*. Although scrambling is sometimes used to mean encryption, we clearly distinguish one from the other. By scrambling, we mean the filtered distortion of the analog output signal. At the receiving end, scrambled analog signal gets unscrambled using the inverse filter

transformation. Scrambling is usually achieved using an analog device to permute the signal in the time domain or to distort the signal in the frequency domain by applying filter banks or frequency converters. Although such transformations are in most cases not secure at all, analog signal scrambling is an interesting case study. Scrambling was simply a product of an immediate industrial need by cable companies for a fast solution to make the free viewing of paid cable channels more difficult than doing nothing at all. The scrambled signal possesses some entertainment value, however, not a great number of people purchased illegal unscrambling cable boxes that were fairly easy to manufacture. One of the reasons might probably be the lack of knowledge on behalf of the ordinary consumers, who did not realize how easy is to break a scrambled signal, or who thought that a cable company can somehow distinguish between those who use legal and those who use illegal unscrambling boxes. But most likely, people just didn't bother. When you really weigh the pros and cons, other than getting some free entertainment, there was no particular value gain. The cost of breaking the encryption system was simply higher than the value of encrypted information. In such cases, scrambling was enough to accomplish the desired effect. On the other hand, there are many multimedia applications where substantial security is of much higher priority. Obviously, using analog scrambling for a secret corporate video message from IBM to Microsoft is a bad idea. Using a conventional cryptosystem in combination with a selective encryption typically provides a much higher level of security.

So far, we have defined what we mean by degradation and scrambling. Both are simply types of lightweight encryption that usually provide a very low level of security. The main difference is that degradation distorts a digital multimedia file, while scrambling distorts an analog multimedia signal. This brings us to another point. In general, the success of scrambling may not guarantee the same success to the digital multimedia degradation. Let us not forget how much cheaper and how much more convenient it is these days to obtain software than it is to obtain hardware. If there is a freeware program that could break a particular degradation method, a consumer can very conveniently download it and use it to break the degraded multimedia. An Internet-based TV channel could be doomed of ever making consumers pay for viewing.

2.5 FORMAT COMPLIANCE

In many applications, it is desired that the encryption algorithm preserve the multimedia format. In other words, after encrypting the encoded multimedia, ordinary decoders can still decode it without crashing. This property of an encryption algorithm is often called *format compliance* (also called *transparency*, *transcodability* or *syntax-awareness*). When feeding the decoder with the format compliant encrypted data, the produced output seems distorted and randomized (see Figure 2.3). If the encryption algorithm was lightweight, the output often gives out some perceptual hints about the original content. However, if the encryption level was substantial, the output usually gives no perceptual information about the original multimedia.

(a) (b) (c)

Figure 2.3. Decoded video produced by an ordinary decoder for (a) video without encryption, (b) video encrypted with lightweight format-compliant encryption, and (c) video encrypted with high-security format-compliant encryption.

Although format compliance is often desired property, for some applications it is not a critical one. In the next few chapters, we discuss both format compliant and non-format compliant multimedia encryption algorithms that are considered important.

2.6 CONSTANT BITRATE AND ERROR-TOLERANCE

In some applications, it is required that the encryption transformation preserves the size of a bitstream. This is known as the *constant bitrate* requirement. However, more often than not it is simply preferred that the output produced by an encryption-equipped encoder and the output produced by an ordinary encoder have similar sizes. That is, the encryption stage is allowed to slightly

increase the size of a bitstream. This is sometimes called a *near-constant bitrate*. A near-constant bitrate is likely to occur when block ciphers are used for encryption, since in that case the encrypted output is always a multiple of the blocksize.

For many multimedia systems *error-tolerance*, or *error-resilience*, is usually of high importance. Since the real-time transport of multimedia data often occurs in noisy environments, which is especially true in the case of wireless channels, the delivered media is prone to bit errors. If a cipher possesses a strong avalanche property, decryption will likely fail even if a single bit is flipped. The error-tolerance can be improved by applying some of the classical error-detecting or error-correcting codes. Unfortunately, these techniques are in many instances extremely costly to apply to an already bulky multimedia bitstream.

SUMMARY

Conventional cryptosystems are powerful but limited in terms of bulky and highly redundant multimedia data. Selective encryption and chaos-based encryption are often used to provide faster and better multimedia security. Various multimedia applications demand different encryption approaches. Consequently, choosing the appropriate security level is often not an easy task. Multimedia specific encryption algorithms are relatively new and the tools for their security assessment are not fully developed. Scrambling refers to a lightweight analog signal distortion, while degradation refers to a lightweight encryption of digital data where the encrypted signal is usually still perceptible. Format-compliance, error-tolerance and constant bitrate are important features of the multimedia encryption algorithms.

Chapter 3

AN OVERVIEW OF MODERN CRYPTOGRAPHY

Cryptography has a rich and astounding history. The first known use of cryptography dates back to ancient Egypt. However, cryptography has gone a long way since then, and modern cryptography represents a much broader and much more complex field. The private sector became interested in cryptography when the computer era began. In the early 1970s, Horst Feistel introduced an important family of ciphers, Feistel ciphers, based on the idea of a product cipher [12]. This led to the creation of the Data Encryption Standard (DES) in 1976 [9], which was the first cryptographic standard adopted by the US Government to provide security for private and business transactions. In that same year, Whitfield Diffie and Martin Hellman pioneered the concept of public-key cryptography in the publication entitled "New Directions in Cryptography" [13]. It is the year when the era of modern cryptography began. In this chapter, along with a brief history, we will discuss various definitions of cryptography, its role, and its fundamental goals. We will also cover some of the basic, yet very important concepts from the modern cryptography.

3.1 A BRIEF HISTORY OF CRYPTOGRAPHY

Before the beginning of the computer era, other than thinking and talking about it as a mysterious and adventurous, ordinary people had little or no interest in cryptography. Throughout history, cryptography was mostly related to military, wars, spies, kings, queens, and pharaohs. Historically, the most popular use of cryptography was to send secret messages. To ensure the secrecy of a message, a sender would either encrypt or hide it. There are numerous documented historical examples of both such methods, and the earliest of them dates back to the ancient Egypt. Ancient Chinese, Greeks, and Romans have used cryptography alike. In fact,

one of the most popular cryptographers of all time was the Roman emperor Julius Caesar.

It is documented that in many of his battles, Caesar would encrypt written messages sent to his centurions by circularly shifting each letter of the Latin alphabet three places to the right. If Caesar had used the English alphabet, he would have encrypted the word "ATTACK" to "DWWDFN". This type of encryption is known as the *Caesar cipher*. In general, if the English alphabet is used, one could use 25 different cyclic shifts to obtain 25 different Caesar-like ciphers. There is a speculation that the name of a famous computer "HAL" from Stanley Kubrick's *2001: A Space Odyssey* was an encryption of the word "IBM", where the alphabet was shifted one place to the left, or equivalently 25 places to the right. Such transformation is part of a larger category of classical ciphers that is referred to as *affine ciphers*. An affine cipher transforms a given letter x into a letter y by using a modular linear bijection. Affine ciphers belong to an even larger category of ciphers called *substitution ciphers*. Amazingly, although almost all substitution ciphers are considered classical ciphers, which are easily broken with the modern mathematics and computational power, the substitution primitives are an integral part of many modern secret-key cryptosystems. Another basic cryptographic primitive, probably just as old and just as important as the substitution cipher, is the *transposition* (or *permutation*) *cipher*. Essentially, the simplest form of a transposition cipher takes a message segment and permutes it internally according to a fixed permutation. When used alone, this type of cipher is very easy to break. However, when used in multiple rounds and in conjunction with a substitution cipher, it constitutes a much more secure *product cipher*, or the *Substitution-Permutation Network (SP-Network)* [5]. One of the earliest known uses of transposition ciphers was in ancient Greece. The *rail fence cipher*, a type of transposition cipher, was used by the Spartans to send messages to Greek warriors. When using a rail fence cipher, the message is continuously written in columns of fixed length, top to bottom. The encrypted message is obtained by reading the text row-wise, left to right. Although the key is the column size, this cipher can be completely represented by a particular permutation. Suppose we would like to encrypt the message "ATTACKTOMORROW" by using rail fence cipher with key 2. Then, we write the text in two rows as follows:

A T C T M R O
T A K O O R W

Finally, by reading the text row by row, we obtain the encrypted message "ATCTMROTAKOORW". As mentioned earlier, this type of transformation can also be described by a permutation, which is in this example (1)(2 8 11 6 10 12 13 7 4 9 5 3)(14). Interestingly, there is a similar movie speculation that involves the transposition cipher. Namely, the speculation is concerning the movie *Sneakers*, in which Len Adleman, one of the co-authors of the famous RSA cryptosystem [17,18], plays a brilliant mathematician that discovers a breakthrough algorithm for integer factorization[1]. The buzz was about the fact that the word "SNEAKERS" could be transposed into "NSAREEKS", where "NSA" stands for the National Security Agency.

As mentioned earlier, a simple transposition cipher is not particularly secure and various statistical techniques can easily break such systems. Moreover, instances of the rail fence cipher, and some other transposition ciphers could possibly have additional deficiencies. Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is that the frist and lsat ltteer be at the rghit pclae. The rset can be a total mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the human mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe. Amzanig, huh?

In Ancient Times, some of the practices used for message hiding were truly fascinating. Herodotus, a famous ancient Greek writer, documented the use of message hiding among the Greeks of the fifth century B.C. In his book *The Histories*, he wrote an interesting story about a Spartan named Demaratus who lived in Persia. When Demaratus found out that the Persian emperor Xerxes was planning to invade Greece, he took a wooden message panel covered in wax used for writing messages at that time, scraped off the wax, wrote a message on the bare wood to Greeks about Xerxes' evil plan, and then covered the wooden panel back with wax. When the innocent looking panel was crossing the border, Persians did not suspect anything. In ancient China, messages were written on pieces of silk that were rolled into little balls and covered in wax. Such balls were

[1] Ironically, such algorithm would break RSA cryptosystem.

swallowed by a courier, and eventually retrieved at the destination, one way or the other.

Cryptographic techniques used for secret communication continued to develop throughout the medieval times, renaissance, and modern period until 1914. However, except for a few hobbyists, cryptography still did not find its use in the private sector and was primarily used for military and government purposes. It was not until the period of World War I and World War II when the use and development of cryptography exploded. In fact, the influence of cryptography and cryptanalysis on the outcome of the two wars was enormous. Militaries and governments continuously exchanged encrypted messages and a successful cryptanalysis often revealed crucial strategic information about the enemy. During WWII, the Germans used a cryptographic machine called *Enigma*, which was in many instances broken by the counterintelligence. Message hiding was brought to a much higher level with the invention of the *microdot*. A page of text was squeezed into a dot of film, one-millimeter in radius, which could be pasted almost undetectably into an ordinary letter.

After World War II a completely new era of communications began. With the introduction of computers and digital messages, not only did military and government need cryptography, but business and private sector did as well. The enormously enhanced computational power brought by computers broke almost all classical ciphers. This is where classical cryptography ends and where modern cryptography begins. An excellent historical reference of cryptography includes David Khan's "The Codebreakers" [14], and, more recently, Simon Singh's "The Code Book" [15].

3.2 SECURE COMMUNICATION

Before we start discussing the definition of cryptography, it is crucial to fully understand the concept of secure communication. In every communication, there are two or more participants. Here, participants refer to people, computers, or anything else capable of conducting communication. Without loss of generality, let us assume a scenario in which there are two communicating participants. Essentially, a communication between the participants consists of exchanging messages over a channel. For example, the channel could be the air, or it could be some kind of computer

network. If two people are talking close to one another, air acts as a channel that transmits the sound waves. Air could be a different kind of channel when wireless technology is used to transmit bits. Let us compare the two. In the case of talking, the channel is quite short since the sound waves could only be carried out a relatively small distance. Unless nearby, or unless a bug is planted on one of the communicants, nobody else can hear the communication. If, however, the two communicants decide to exchange written messages instead of words, planted bugs or nearby eavesdropping is useless. There are other ways one can still spy on the conversation, but such ways are usually too complex and inconvenient to accomplish. In the second case, where the communication is carried using a wireless data exchange, the channel could be enormously large. For example, if two people are talking over their cell phones, the channel could range over large areas, even different countries. Preventing eavesdropping of such communications is much harder, if not impossible. For the most part, computer networks belong to this group of communication channels. Suppose two participants, Alice and Bob[2], are communicating over a channel. Alice and Bob would like to communicate confidentially, in other words, nobody else should be able to read or hear the exchanged messages by eavesdropping the channel. Unless they manage to disguise the messages that are being sent through the channel, an eavesdropper could comprehend them. Fortunately, cryptography solves the problem of conducting confidential communications.

The concept of secure communications can be extended. In many instances, people wish to secure their own confidential information for later access. For example, suppose Alice would like to store her personal files in a secure place. One can think of this situation as a case of the communication security. Communication is conducted between Alice at time t_0 and Alice at later time t_1. In a situation like this, the channel of the communication is the storage device that Alice uses to store her personal files between times t_0 and t_1. Security for this scenario can be achieved the same way it is achieved in case of classical communication.

---

[2] Cryptographers often use the names Alice and Bob for two communicating parties, which is certainly much friendlier than calling them person *A* and person *B*.

Finally, there are situations where a larger group of people is communicating together. However, the system could be dynamic in the sense that people can join the group, leave the group, or they can be banned from the group during the communication. A real life example of this setting could be a business audioconferencing meeting where multiple participants join and leave the meeting. This is referred to as the *secure group communication*. For example, when Bob joins the group, he must be able to hear the exchanged group messages only from that point in time until the time when he leaves the group. He must not be able to access the messages that were exchanged before him joining the group. In addition, Bob must not be able to access the messages that are exchanged between the rest of the group after he leaves the group, or after the group members ban him out of the group. Many security protocols provide a solution to this and similar types of group communication scenarios. The details of secure group communication are beyond the scope of this book, but an excellent reference book that covers this subject is "Secure Group Communications over Data Networks" by X. Zou, B. Ramamurthy, and S. Magliveras [16].

To ensure secure communication, cryptography essentially transforms and disguises the messages that are being sent over a communication channel. One can apply the transformation to the original message, or the *plaintext*, to obtain the transformed message, or the *ciphertext*. This process is called *encryption*. Analyzing the ciphertext must not reveal the corresponding plaintext. An important property of this kind of transformation is that it must be invertible, but unless a particular secret is known, getting the plaintext directly from the ciphertext is hard. This secret is often referred to as the *key*. Finally, the inverse transformation is called the *decryption*.

Therefore, to conduct a secure communication, Alice and Bob have to encrypt their messages before sending them over the channel. Upon receiving the encrypted messages, Alice and Bob are able to recover the plaintext using the secret key. An eavesdropper does not know the secret key, and in that case the decryption is difficult. This is the main principle of secure communication where cryptography is used to ensure confidentiality. This is also what cryptography was all about until the 1970s. Since then, it has boomed into a much broader subject.

3.3 DEFINITION OF CRYPTOGRAPHY

Cryptography, to some people's surprise, is not the most general discipline. Cryptology is. *Cryptology* refers to an umbrella under which cryptography, cryptanalysis and steganography are studied. Sometimes, cryptology only represents cryptography and cryptanalysis, and many times cryptography is used to mean cryptology. In that respect, it is important that we clarify the meaning of cryptography, cryptanalysis and steganography for the scope of this book.

The word cryptography originates from the Greek words κρυπτος (kriptos) meaning hidden, and χραφη (grafee) meaning writing. Therefore, cryptography literally refers to the study of hidden writing. In actuality, cryptography is not easy to define. Cryptography is often defined as the science of conducting secure communication. However, that is only partly true. While one of the main goals of cryptography is to provide confidentiality during the communication, cryptography seeks to provide solutions to other aspects of information security, such as data integrity, authentication, and non-repudiation. Therefore, we define *cryptography* as an art of manipulating the information in order to accomplish all information security objectives. *Confidentiality*, or *privacy*, refers to the protection of information from unauthorized access. An undesired communicating party (called *adversary*) must not be able to access the communication material. *Data integrity* ensures that information has not been manipulated in an unauthorized way. *Authentication* studies two different concepts: *entity authentication* and *message authentication*. Message authentication provides assurance of the identity of the message sender. This type of authentication also includes an evidence of data integrity. If the message is modified during transmission, the sender cannot be the originator of the message. Entity authentication assures the receiver of a message of both the identity of the sender and his active participation. Finally, *non-repudiation* refers to preventing the denial of previously executed actions.

While cryptography is trying to provide information security, cryptanalysis is trying to damage it. *Cryptanalysis* can be defined as a study of techniques used for deliberate destruction of information security provided by cryptography. For example, cryptanalysis seeks to provide methods for recovering an encrypted message in whole, or

in part, when the decryption key is not known. Depending on the amount of known information and the amount of control over the system by the adversary, there are several basic types of cryptanalytic attacks. In a *ciphertext-only attack*, the adversary only has access to one or more encrypted messages. The most important goal of a proposed cryptosystem is to withstand this type of attack. An important instance of ciphertext-only attack is the *brute force attack*. This type of attack is based on an exhaustive key search, and for a well-designed cryptosystem, it should be computationally infeasible. In a *known-plaintext attack*, the adversary has some knowledge about the plaintext corresponding to the given ciphertext. This may help determine the key or a part of the key, which ultimately could result in recovering the entire set of messages that were encrypted using that particular key. In the *chosen-plaintext attack*, the adversary can feed the chosen plaintext into the black box that contains the encryption algorithm and the encryption key. The black box spits out the corresponding ciphertext, and the adversary may use the accumulated knowledge about the plaintext-ciphertext pairs to obtain the secret key or at least a part of it. At last, the *chosen-ciphertext attack* refers to a cryptanalytic attack where an adversary can feed the chosen ciphertext into the black box that contains the decryption algorithm and the decryption key. The black box produces the corresponding plaintext, and the adversary tries to obtain the secret key or a part of the key by analyzing the accumulated ciphertext-plaintext pairs.

Steganography essentially refers to a hiding of the communication channel. It originates from the ancient Greek words στεγανος (steganos) meaning cover or seal, and χραφη (grafee) meaning writing. This probably comes from the Herodotus' story about the covering of secret messages written on a wooden panel with wax. Data hiding is often used as a synonym for steganography. Cryptography could be used in combination with steganography, which adds an additional layer of security. While cryptography was a field often based on complex mathematics and abstract logic, steganography was always more or less relying on the use of a concrete material from the physical world. As such, steganography was probably used more often than cryptography throughout history. Another reason for this is that even an extremely simple form of steganography could be surprisingly effective in practice. If Alice sends to Bob a message that looks like gibberish, an eavesdropper presumes that the message has been encrypted.

However, if Alice sends to Bob a message "HAPPY BIRTHDAY", which is actually a steganography related code for "ATTACK TOMORROW", an eavesdropper may not suspect a thing.

3.4 THE ROLE OF CRYPTOGRAPHY IN SECURE SYSTEMS

In many security systems, cryptography is a major factor. However, it is important to understand that the role of cryptography is not to secure the system. The role of cryptography is to help securing the system. Having a well-designed cryptography in place does not necessarily ensure that the entire system is secure. Most of the time, a particular cryptosystem has been previously specified, and in that respect, cryptography represents an easy part in the secure system design. However, the way in which cryptography is implemented and the way in which it is integrated into the entire system determines whether the system is secure or not. For that reason, complex secure systems are often extremely difficult to design. In most cases, the real world attacks on security systems exploit incorrect implementation and false assumptions. There are few real-life examples of successful attacks on the cryptosystem itself. Unfortunately, security of the entire system is in jeopardy when only one weakness exists. Consequently, even a well-designed cryptography could easily fail to be effective.

3.5 BASIC CONCEPTS FROM MODERN CRYPTOGRAPHY

Although we have already used the word "cryptosystem", we now define it more formally. In modern cryptography, a *cryptosystem* includes the exact specification of the encryption and decryption algorithms, the key size(s) for which these algorithms were designed, as well as all other parameters needed for setup, initialization or input related to these algorithms. *Cipher* is very often used as a synonym for cryptosystem. All cryptosystems (ciphers), for the most part, follow a set of requirements called *Kerckhoffs' desiderata* published by A. Kerckhoffs back in 1883 [28]. One of the most important requirements, often referred to as the *Kerckhoffs' principle*, from the Kerckhoffs' desiderata is that the algorithms for encryption and decryption within a cryptosystem must be publicly available. In other words, no cryptosystem should rely on the

secrecy of its encryption or decryption algorithm. Consequently, its security should rely entirely upon the secret key.

The importance of the Kerckhoffs' principle lies in the fact that keeping the encryption or decryption algorithm secret is difficult. These algorithms do not usually change significantly over a long period of time, and many software or hardware implementations that are available to users could be reverse-engineered. Even worse, not only could an adversary be one of the cryptosystem users, but there could be adversaries among the people that designed or implemented it. Information could easily leak from one source or other, and the secrecy of all messages ever protected by that cryptosystem would be in jeopardy, provided that the secrecy of the encryption or decryption algorithm is a crucial component of the entire system. Another reason for the validity of Kerckhoffs' principle is that if an algorithm is made public then the expert community could evaluate its design. It is often easy to make small design mistakes that could make the entire cryptosystem utterly useless. That is why a conventional cryptosystem is one that had been investigated by the cryptographic community over a longer period of time. Next, we define different types of cryptosystems (ciphers).

3.5.1 Private-Key Cryptography

All *private-key (symmetric) cryptosystems* have a common property: they rely on a shared secret between communicating parties. This secret is used both as an encryption key and as a decryption key (thus the keyword "symmetric" in the name). This type of cryptography ensures only confidentiality and fails to provide the other objectives of information security. Another important deficiency of symmetric-key cryptosystems is that they cannot handle large communication networks. If a node in a communication network of n nodes needs to communicate confidentially with all other nodes in the network, it needs $n - 1$ shared secrets. For large values of n, this is highly impractical and inconvenient.

On the other hand, an advantage over public-key cryptosystems is that symmetric cryptosystems require much smaller key sizes for the same level of security. Hence, computations are much faster and the memory requirements are smaller. All classical cryptosystems

are examples of symmetric-key cryptosystems. In addition, most modern cryptosystems are symmetric as well. Some of the most popular examples of modern symmetric-key cryptosystems include AES (Advanced Encryption Standard) [11], DES (Data Encryption Standard) [9], IDEA [10], and many others.

3.5.2 Public-Key Cryptography

Public-key cryptosystems, also called *asymmetric-key cryptosystems*, contain two different keys: a *public key*, which is publicly known, and the *secret key*, which is kept secret by the owner. The system is also called "asymmetric" since different keys are used for the encryption and decryption. If data is encrypted with a public key, it can only be decrypted using the corresponding private key, and vice versa. Today, all public-key cryptosystems rely on some computationally intractable problem. For example, the cryptosystem RSA [17,18] relies on the difficulty of factoring large integers, while the El-Gamal [19] cryptosystem relies on the discrete logarithm problem (DLP), which is the problem of finding a logarithm of a group element with generator base in a finite Abelian group.

Public-key cryptosystems do not need to have a shared secret between the communicating parties. This solves the problem of a large confidential communication network introduced earlier. In addition, public-key cryptography enables ways of achieving all goals of cryptography. By means of combining public-key cryptography, public-key certification, and secure hash functions, there are protocols that enable digital signatures, authentication, and data integrity. Due to the increase in processor speed and the advances in modern cryptanalysis, the key size used in the modern public-key cryptosystems became very large. As an example, a 128-bit key used with DES cryptosystem has approximately the same level of security as the 1024-bit key used with RSA public-key cryptosystem [2]. This created a disadvantage in comparison with the secret-key cryptosystems. Namely, the public-key cryptography is significantly slower, and it requires a large memory capacity and computational power.

To solve these problems, researchers introduced various approaches. In order to decrease the key size so that public-key cryptography can be used in the smaller computational environments (such as smart cards or handheld wireless devices), in 1985 Neil Koblitz [20] and

Victor Miller [21] independently introduced the idea of using a more complex elliptic curve group as an underlying public-key algebraic structure. The elliptic curve cryptography enables smaller key sizes since the discrete logarithm problem is much harder to solve in the elliptic curve group. Another strategy was to design a public-key cryptosystem that relies on a more complex computational problem, such as the lattice reduction problem. The relatively new cryptosystem NTRU, introduced by Hoffstein, Pipher and Silverman in [22], is based on the algebra of a ring of truncated polynomials. The system relies on the lattice reduction problem, and it represents the only public-key cryptosystem that has the speed, memory, and computational complexity comparable to that of a secret-key cryptosystem. However, since NTRU was proposed relatively recently, its security aspects are yet to be investigated [23,24]. The most common solution to improving the speed of a public-key system is to create a hybrid scheme by combining a symmetric-key with a public-key cryptosystem. Namely, the plaintext is encrypted using a fast symmetric-key scheme, and only the secret key used for the symmetric encryption is encrypted with a slow public-key scheme such as RSA. This way all goals of cryptography can be achieved with a much better performance.

3.5.3 Block Ciphers

A *block cipher* is a cryptosystem where encryption and decryption functions map n-bit input into an n-bit output. For each block cipher, there are different *modes of operation* available. Two most popular modes of operation include the *electronic codebook* (ECB) mode, and the *cipher-block chaining* (CBC) mode. In the ECB mode, the entire message (plaintext) is divided into blocks of size n, and then each block is encrypted using the encryption function. If the number of bits in the message is not an exact multiple of n, the message is usually padded. The result of this is concatenation of n-bit ciphertext blocks. The decryption function takes these ciphertext blocks and decrypts them one at a time. The parameter n is often denoted by *blocksize*, or *blocklength*. Block ciphers usually have blocklength of 64-bits or more. Most available cryptosystems, both symmetric and asymmetric, are block ciphers. Note that a block cipher can be thought of as a simple substitution cipher, such as the Caesar cipher, in which the number of symbols is extremely large. More precisely, the number of substitution symbols is 2^n, which is a large number when $n \geq 64$.

In a block cipher, each block is transformed according to a proper s-bit secret key k (see Figure 3.1). If the key is invalid, the transformation fails to produce correct n-bit output. In the case of decryption, it fails to recover a message. Although multiple keys for multiple blocks or groups of blocks could be used, it is usually sufficient to use the same key in the transformation of each block.

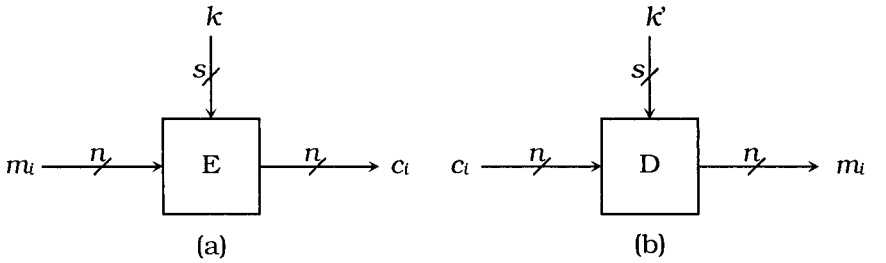


Figure 3.1. (a) Encryption, and (b) decryption transformation of the i^{th} block within a block cipher in ECB mode. This scheme is applicable to both symmetric and asymmetric block ciphers, however, k' = k if a symmetric-key cipher is used, and k' ≠ k if an public-key cipher is used.

In such a setting, exhaustive key search depends on the key size s. In fact, the key space is 2^s, so in the worst case scenario an attacker must try 2^s different keys in order to find the secret key. Although an attacker would only need to test up to 2^s keys, the likelihood of finding the key in the feasible time is nearly zero for sufficiently large value of s. Therefore, s is usually picked to be at least 64, but larger values of s such as 128 or 256 are recommended to ensure higher security.

The ECB mode is suitable for noisy channels, since the errors do not propagate to multiple blocks. Namely, if an error occurs within a ciphertext block c_i, this only affects the recovery of a corresponding plaintext block m_i. However, block ciphers that run in ECB mode do preserve the identical block patterns, and identical plaintext blocks are encrypted into identical ciphertext blocks. That is, if $m_i = m_j$, then $c_i = c_j$. Consequently, the ECB mode is not recommended for encrypting long messages or data with high redundancy such as images or videos.

In the CBC mode of operation (Figure 3.2), the message block is XOR-ed with the previously encrypted block, which results in a stronger encryption effect for the redundancy-rich data. At the beginning of both encryption and decryption process, the first message block is XOR-ed with some predefined initialization vector c_0.

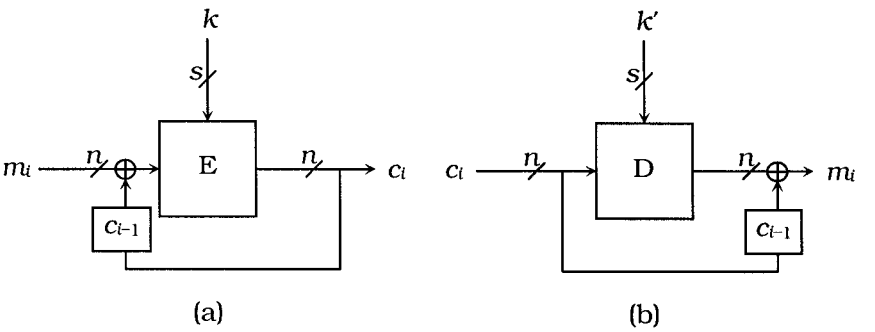


Figure 3.2. (a) Encryption, and (b) decryption transformation of the i^{th} block within a block cipher in CBC mode. If a symmetric-key cipher is used then k' = k, and if an asymmetric-key cipher is used then k' ≠ k.

Unfortunately, although a better encryption effect is achieved for redundant data, the CBC mode has weaker error tolerance. If an error occurs within the ciphertext block c_i, the recovery of plaintext blocks m_i and m_{i+1} is affected. In the case of multimedia data such as images and videos, we highly recommend not using the ECB mode at all. Instead, one should use one of the more secure and complex modes, such as the CBC mode. It is worth mentioning here that other more complex modes of operation could be used to practically turn a given block cipher into a stream cipher.

3.5.4 Stream Ciphers

Stream ciphers are a very important group of ciphers. A *stream cipher* is a cryptosystem that transforms (encrypts or decrypts) individual symbols of the alphabet, which are usually simply binary digits, one at a time. However, the transformation often varies in time. This is a different approach than the approach used by the block ciphers, where a fixed transformation is applied to a group of

symbols, usually 64 or more at a time. Although some modes of operation related to block ciphers operate in a stream cipher fashion, the distinction between these two groups of ciphers should be clear.

In general, stream ciphers are very suitable for hardware implementation. If implemented in hardware, stream ciphers have relatively simple architecture that makes them exceptionally faster than the block ciphers. Applications that require extremely fast encryption processing, such as many multimedia applications, could include a dedicated hardware unit that is based on a stream cipher. Since stream ciphers operate on a small amount of data, usually bits, they are also appropriate for applications where bits or characters are received one at a time, or where the buffer size is limited. Another advantage of stream ciphers is the error resilience, since small amounts of bits are processed at a time and no error propagation occurs. This is very suitable when the transmission is carried over a noisy channel.

For a given secret key k, a stream cipher generates a pseudorandom sequence of symbols. Thus, one can think of the stream cipher as a continuous random symbol generator where a random seed is either k, or it depends on k. The generated sequence of symbols is called a *keystream*. In any stream cipher, the same alphabet (set of symbols) is used to represent plaintext, ciphertext and keystream. In most cases, the alphabet consists of binary digits and a stream cipher generates a pseudorandom sequence of bits. Encryption is achieved by applying modular addition of plaintext symbols with keystream symbols where the modulus is the size of alphabet. Naturally, decryption is realized by subtracting the keystream symbols from the ciphertext symbols modulo the size of alphabet. In the case where the alphabet consists only of binary digits (bits), both operations are equivalent to bitwise XOR (see Figure 3.3).

There are two groups of stream ciphers depending on whether the bit generator seed is k, or it is a combination of key k and some previous segments of plaintext. *Synchronous stream ciphers* are those in which the keystream only depends on the secret key. On the other hand, *self-synchronizing* (or *asynchronous*) *stream ciphers* generate the keystream depending on both the key and a fixed number of previous plaintext symbols. The technique of utilizing some previous parts of the input to confuse the data is often referred to as *chaining*.

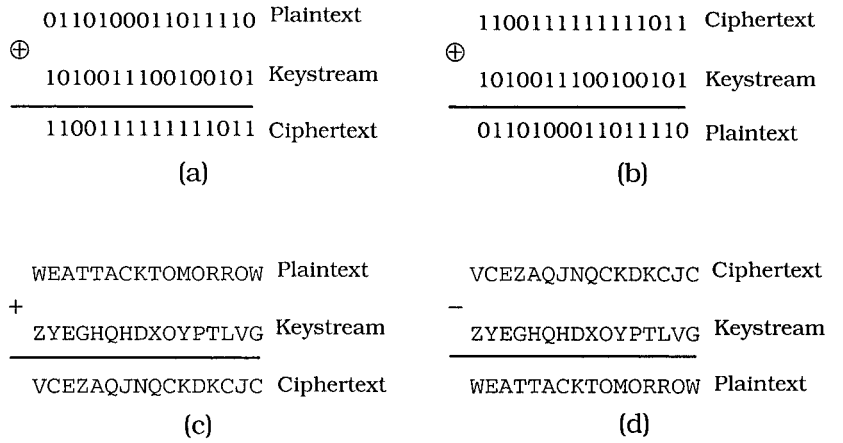


Figure 3.3. (a) Encryption and (b) decryption using a stream cipher with binary alphabet, and (c) encryption and (d) decryption using a stream cipher with English alphabet

3.5.5 Vernam Cipher and One-Time Pad

Using the keystream that is as long as the message was first introduced by G.S. Vernam in the early 20th century. The *Vernam cipher* (that was patented by Vernam in 1919 [25]) operates on the binary alphabet and it consists of using a binary key, which is as long as the binary message. The encryption and decryption transformations are identical to those used within the binary stream cipher (see Figure 3.3-a and Figure 3.3-b), except that the keystream is not generated from the key, but the key is the keystream itself.

A type of Vernam cipher where the key is truly random and used only once is known as a *one-time pad.* This is the only theoretically unbreakable cipher that people have ever used. Shannon proved that a one-time pad is unconditionally secure, that is, the ciphertext-only attack is not possible since the ciphertext is uniformly distributed and completely uncorrelated. However, there are some major practical drawbacks of a one-time pad. The truly random sequence is hard to generate. Many pseudorandom number generators exist, but the word "pseudo" indicates that the sequence is not truly random. There are some techniques that people have utilized, such as the ones based on detecting the number or velocity

of atomic particles in the environment, but such techniques are expensive to implement. Even if the random sequence is obtained, Alice has to securely transmit this sequence to Bob, but using every other encryption in doing so kills off the unconditional security aspect. Thus, such a sequence would have to be physically transmitted to Bob, possibly using a trusted courier. This must be done for every communicant, and for every single message. In addition, the random sequence must be long enough to facilitate encryption of an entire message. All this makes implementing a one-time pad difficult in practice. It has been reported that a one-time pad was used for securing Washington's top-secret communication with Moscow during the cold war. During this time, a trusted courier was used to transmit the random sequence (the key).

Clearly, this type of encryption is not suitable for the private sector. One-time pad encryption would be impossible to implement and maintain for larger communication networks. In respect to multimedia encryption, it would be impractical and inefficient to implement a costly one-time pad for securing the transmission of relatively low-value multimedia content. Instead of one-time pad and Vernam cipher, a more suitable approach would be to use a stream cipher with a given secret key of limited size, and generate a pseudorandom sequence to be used as a keystream. A fast dedicated hardware implementation of a stream cipher would make it applicable to many demanding multimedia applications.

3.5.6 Hash Functions

Hash functions are extremely useful transformations in modern cryptography. They essentially serve to obtain a *digest* (also called *digital fingerprint*, *hash value*, or *hash*) of a message. Generally speaking, *hash functions* are transformations that take information of arbitrary length and produce an output of some fixed length. Hash functions are used in many non-cryptographic applications, but hash functions that are used in cryptography are subject to additional constraints. A cryptographic hash function h must be such that for any pre-specified output y it is computationally infeasible to obtain the value x for which $h(x) = y$. Formally, this is referred to as the *preimage resistance*. In addition, h should be such that for a given x and $y = h(x)$ it is computationally infeasible to find a second input x', different from x, for which $h(x') = y$. This is known as the *second-preimage resistance*. Clearly since the domain of h is

larger than the range of h, there are collisions; i.e., there are (usually many) values x and x' such that $x' \neq x$ and $h(x) = h(x')$. A cryptographically good hash function h should be such that it is computationally hard to obtain a random pair x and x' such that $x \neq x'$ and $h(x) = h(x')$. If this criterion is satisfied, it is said that h is *collision resistant*. In fact, collision resistance implies the second-preimage resistance, so that if h is collision resistant it is also second-preimage resistant. In many applications, additional constraints are imposed, such as non-correlation, near-collision resistance and partial-preimage resistance. *Non-correlation* simply requires that input bits and output bits must not be statistically correlated, while *partial-preimage resistance* requires that it must be equally difficult to obtain any part of the input as it is to obtain the entire input. Near-collision resistance is achieved when it is computationally infeasible to find pairs x and x' for which $h(x)$ and $h(x')$ differ by a small Hamming distance.

There are two distinct groups of cryptographic hash functions. The first group, referred to as the *keyed hash functions*, consists of hash functions that in addition to the message require a secret key as the input. As such, they are mostly used for providing message authentication and integrity in many security protocols. Important examples of keyed hash functions are well-studied *message authentication codes* (MACs). The second group of cryptographic hash functions, so-called unkeyed hash functions, have only message as the input parameter. Unkeyed cryptographic hash functions represent an essential ingredient of a digital signature scheme. They are used to obtain a message digest (its digital fingerprint) that is authenticated using public-key cryptography. The most widely used hash functions are Secure Hash Algorithm-1 (SHA-1), SHA-256, and recently also SHA-384 and SHA-512. These are a part of the adopted standard called Secure Hash Standard (SHS) [26]. Other popular modern unkeyed cryptographic hash functions include, Message Digest 4 and 5 (MD4 and MD5), and RIPEMD-160.

3.5.7 Digital Signatures

The number of applications for digital signatures is enormous. A digital signature allows us to electronically sign a legal document, and in many instances, it provides better authentication than the old-fashioned ink-on-paper signature. In other words, it is generally

much more difficult to forge a properly implemented digital signature than it is to forge a handwritten ink signature. In terms of information security, digital signatures provide means of authentication, data integrity and non-repudiation.

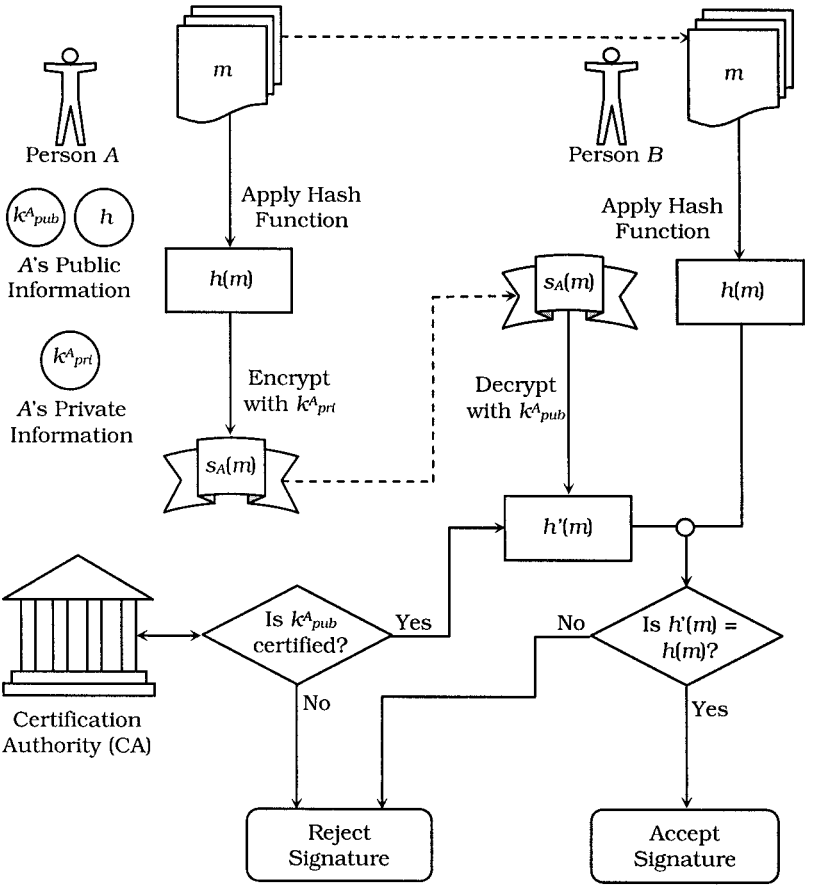


Figure 3.4. General model for a digital signature scheme: person *A* signs a message *m* and person *B* checks the validity of the signature.

In general, a digital signature scheme is based on a public-key cryptosystem and a secure hash function. For a public-key cryptosystem *c*, and a user *A*, we define k^A_{pub} to be *A*'s public key known to everybody and k^A_{prt} to be *A*'s private key known only to *A*.

Let $c(m,k)$ denote a cryptographic transformation of message m by a cryptosystem c with a key k. Notice that for every public-key cryptosystem c, the following equation holds:

$$c(c(m,k^A_{pri}),k^A_{pub}) = c(c(m,k^A_{pub}),k^A_{pri}) = m.$$

Also, let h denote a secure hash function (such as SHA-1 [26]). Then, a digital signature of a person A for a given document m, denoted by $s_A(m)$, can be realized by the following transformation:

$$s_A(m) = c(h(m),k^A_{pri}).$$

Notice that $s_A(m)$ can only be generated by person A, since k^A_{pri} is known only to A. For a given m and $s_A(m)$ everybody can verify the validity of a signature by computing the following two values $c(s_A(m), k^A_{pub})$ and $h(m)$, and comparing them. If the two values are equal then the signature is valid. Otherwise, it is an invalid signature and the document is marked as illegal. The general model for digital signatures is presented in Figure 3.4. Interestingly, the public keys are often properly verified and then certified by the *Certification Authority* (CA) by using a digital signature scheme itself. The Certification Authority digitally signs valid public keys, and the signature, in essence, acts as a digital certificate.

As mentioned earlier, a digital signature scheme, which can be denoted as a pair (c,h), provides authentication, data integrity and non-repudiation. The authentication refers to determining the identity of a signer. Data integrity ensures that modifying the content of a document, even in the slightest way, makes the signature illegal. Finally, the non-repudiation eliminates the false claim by person A about not signing a particular document when A did indeed sign it.

The Government has adopted the standard called *Digital Signature Standard* (DSS) [27]. This standard uses the SHA-1 hash function together with a one of the following three public-key cryptosystems: RSA, Digital Signature Algorithm (DSA), which is a slight variation of El-Gamal cryptosystem, and ECDSA (elliptic curve-based DSA) [27]. In the next chapter, we present the details of some of the most important public-key cryptosystems that could be utilized in the digital signature model from the above.

SUMMARY

Cryptography has been used throughout history; however, the business and private sector have only become interested in cryptography since the boom of computers and electronic communications. Modern cryptography started in the 1970s with the idea of a public-key cryptosystem that enabled the implementation of the major goals of cryptography: confidentiality, authentication, data integrity and non-repudiation. The important concepts from modern cryptography include public-key (asymmetric-key) cryptography, secret-key (symmetric-key) cryptography, block ciphers, stream ciphers, secure hash functions, and digital signatures. The U.S. Department of Commerce and U.S. National Institute of Standards and Technology (NIST) have adopted many standards to allow secure transactions within the business and private sector.

Chapter 4

IMPORTANT MODERN CRYPTOSYSTEMS

Some of the most important and most widely used conventional symmetric-key and public-key cryptosystems are presented in this chapter. We describe the fundamentals of the DES, IDEA, AES, RSA, ElGamal and Diffie-Hellman protocol. As we shall later see, all of these well-established ciphers and protocols could be used as an underlying basis for many selective encryption techniques specifically designed for multimedia.

4.1 DATA ENCRYPTION STANDARD (DES)

International Business Machines (IBM) submitted Data Encryption Standard (DES) to the U.S. National Bureau of Standards in 1974, and in November of 1976 it became a federal standard. DES was finalized in the 1977 U.S. Federal Information Processing Standard (FIPS) publication 46 [9]. Although some attacks on DES were discovered since its release (such as the differential and linear cryptanalysis), DES remains moderately strong and widely used in many commercial applications.

DES is a symmetric block cipher where the input and output blocksize is 64 bits. The key size is 64 bits, however, eight of them may be used for parity, which reduces the effective key size to 56 bits. The algorithm consists of 16 rounds and each round uses a 48-bit subkey K_i, $1 \leq i \leq 16$, computed by the DES Key Schedule Algorithm. DES algorithm makes use of three permutation functions denoted by IP, IP^{-1}, and P, one expansion function denoted by E, and eight S-boxes (substitution boxes). Here, IP^{-1} is an inverse permutation of IP.

$IP([x_1,\ldots,x_{64}]) = [x_{58}, x_{50}, x_{42}, x_{34}, x_{26}, x_{18}, x_{10}, x_2, x_{60}, x_{52}, x_{44}, x_{36}, x_{28}, x_{20}, x_{12}, x_4, x_{62}, x_{54}, x_{46}, x_{38}, x_{30}, x_{22}, x_{14}, x_6, x_{64}, x_{56}, x_{48}, x_{40}, x_{32}, x_{24}, x_{16}, x_8, x_{57},$

$x_{49}, x_{41}, x_{33}, x_{25}, x_{17}, x_9, x_1, x_{59}, x_{51}, x_{43}, x_{35}, x_{27}, x_{19}, x_{11}, x_3, x_{61}, x_{53}, x_{45}, x_{37}, x_{29}, x_{21}, x_{13}, x_5, x_{63}, x_{55}, x_{47}, x_{39}, x_{31}, x_{23}, x_{15}, x_7]$,

$IP^{-1}([x_1,\ldots,x_{64}]) = [x_{40}, x_8, x_{48}, x_{16}, x_{56}, x_{24}, x_{64}, x_{32}, x_{39}, x_7, x_{47}, x_{15}, x_{55}, x_{23}, x_{63}, x_{31}, x_{38}, x_6, x_{46}, x_{14}, x_{54}, x_{22}, x_{62}, x_{30}, x_{37}, x_5, x_{45}, x_{13}, x_{53}, x_{21}, x_{61}, x_{29}, x_{36}, x_4, x_{44}, x_{12}, x_{52}, x_{20}, x_{60}, x_{28}, x_{35}, x_3, x_{43}, x_{11}, x_{51}, x_{19}, x_{59}, x_{27}, x_{34}, x_2, x_{42}, x_{10}, x_{50}, x_{18}, x_{58}, x_{26}, x_{33}, x_1, x_{41}, x_9, x_{49}, x_{17}, x_{57}, x_{25}]$,

$P([x_1,\ldots,x_{32}]) = [x_{16}, x_7, x_{20}, x_{21}, x_{29}, x_{12}, x_{28}, x_{17}, x_1, x_{15}, x_{23}, x_{26}, x_5, x_{18}, x_{31}, x_{10}, x_2, x_8, x_{24}, x_{14}, x_{32}, x_{27}, x_3, x_9, x_{19}, x_{13}, x_{30}, x_6, x_{22}, x_{11}, x_4, x_{25}]$,

$E([x_1,\ldots,x_{32}]) = [x_{32}, x_1, x_2, x_3, x_4, x_5, x_4, x_5, x_6, x_7, x_8, x_9, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{16}, x_{17}, x_{18}, x_{19}, x_{20}, x_{21}, x_{20}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_1]$.

The S-boxes, denoted by $S_1,\ldots,S_8$, map a 6-bit input into a 4-bit output. They are fixed and defined as follows:

X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S_1(x)$	14	0	4	15	13	7	1	4	2	14	15	2	11	13	8	1
$S_2(x)$	15	3	1	13	8	4	14	7	6	15	11	2	3	8	4	14
$S_3(x)$	10	13	0	7	9	0	14	9	6	3	3	4	15	6	5	10
$S_4(x)$	7	13	13	8	14	11	3	5	0	6	6	15	9	0	10	3
$S_5(x)$	2	14	12	11	4	2	1	12	7	4	10	7	11	13	6	1
$S_6(x)$	12	10	1	15	10	4	15	2	9	7	2	12	6	9	8	5
$S_7(x)$	4	13	11	0	2	11	14	7	15	4	0	9	8	1	13	10
$S_8(x)$	13	1	2	15	8	13	4	8	6	10	15	3	11	7	1	4

X	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$S_1(x)$	3	10	10	6	6	12	12	11	5	9	9	5	0	3	7	8
$S_2(x)$	9	12	7	0	2	1	13	10	12	6	0	9	5	11	10	5
$S_3(x)$	1	2	13	8	12	5	7	14	11	12	4	11	2	15	8	1
$S_4(x)$	1	4	2	7	8	2	5	12	11	1	12	10	4	14	15	9
$S_5(x)$	8	5	5	0	3	15	15	10	13	3	0	9	14	8	9	6
$S_6(x)$	0	6	13	1	3	13	4	14	14	0	7	11	5	3	11	8
$S_7(x)$	3	14	12	3	9	5	7	12	5	2	10	15	6	8	1	6
$S_8(x)$	10	12	9	5	3	6	14	11	5	0	0	14	12	9	7	2

x	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$S_1(x)$	4	15	1	12	14	8	8	2	13	4	6	9	2	1	11	7
$S_2(x)$	0	13	14	8	7	10	11	1	10	3	4	15	13	4	1	2
$S_3(x)$	13	1	6	10	4	13	9	0	8	6	15	9	3	8	0	7
$S_4(x)$	10	3	6	15	9	0	0	6	12	10	11	1	7	13	13	8
$S_5(x)$	4	11	2	8	1	12	11	7	10	1	13	14	7	2	8	13
$S_6(x)$	9	4	14	3	15	2	5	12	2	9	8	5	12	15	3	10
$S_7(x)$	1	6	4	11	11	13	13	8	12	1	3	4	7	10	14	7
$S_8(x)$	7	2	11	1	4	14	1	7	9	4	12	10	14	8	2	13

x	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$S_1(x)$	15	5	12	11	9	3	7	14	3	10	10	0	5	6	0	13
$S_2(x)$	5	11	8	6	12	7	6	12	9	0	3	5	2	14	15	9
$S_3(x)$	11	4	1	15	2	14	12	3	5	11	10	5	14	2	7	12
$S_4(x)$	15	9	1	4	3	5	14	11	5	12	2	7	8	2	4	14
$S_5(x)$	15	6	9	15	12	0	5	9	6	10	3	4	0	5	14	3
$S_6(x)$	7	11	0	14	4	1	10	7	1	6	13	0	11	8	6	13
$S_7(x)$	10	9	15	5	6	0	8	15	0	14	5	2	9	3	2	12
$S_8(x)$	0	15	6	12	10	9	13	0	15	3	3	5	5	6	8	11

The S-boxes can be viewed as a single transformation, denoted by S, that maps a *48*-bit input into a *32*-bit output by sequentially applying the transformations S_i, $1 \le i \le 8$ (see Figure 4.1).

In addition, the DES Key Schedule Algorithm uses two transformations, PC_1 and PC_2. The former is used to select 56 bits in a particular order out of a 64-bit input, while the later is used to select 48 bits in a particular order out of a 56-bit input:

$PC_1([x_{1,\ldots,}x_{64}]) = [x_{57}, x_{49}, x_{41}, x_{33}, x_{25}, x_{17}, x_9, x_1, x_{58}, x_{50}, x_{42}, x_{34}, x_{26}, x_{18}, x_{10}, x_2, x_{59}, x_{51}, x_{43}, x_{35}, x_{27}, x_{19}, x_{11}, x_3, x_{60}, x_{52}, x_{44}, x_{36}, x_{63}, x_{55}, x_{47}, x_{39}, x_{31}, x_{23}, x_{15}, x_7, x_{62}, x_{54}, x_{46}, x_{38}, x_{30}, x_{22}, x_{14}, x_6, x_{61}, x_{53}, x_{45}, x_{37}, x_{29}, x_{21}, x_{13}, x_5, x_{28}, x_{20}, x_{12}, x_4]$,

$PC2([x_1,...,x_{56}]) = [x_{14}, x_{17}, x_{11}, x_{24}, x_1, x_5, x_3, x_{28}, x_{15}, x_6, x_{21}, x_{10}, x_{23}, x_{19}, x_{12}, x_4, x_{26}, x_8, x_{16}, x_7, x_{27}, x_{20}, x_{13}, x_2, x_{41}, x_{52}, x_{31}, x_{37}, x_{47}, x_{55}, x_{30}, x_{40}, x_{51}, x_{45}, x_{33}, x_{48}, x_{44}, x_{49}, x_{39}, x_{56}, x_{34}, x_{53}, x_{46}, x_{42}, x_{50}, x_{36}, x_{29}, x_{32}]$.

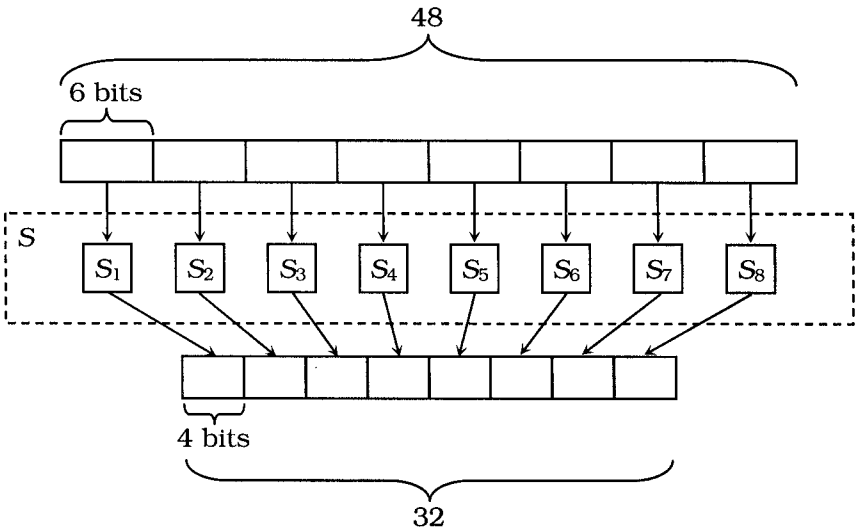


Figure 4.1 The S transformation that maps a 48-bit input into a 32-bit output according to the S-boxes.

Next, we present the details of both DES Encryption Algorithm and DES Key Schedule Algorithm.

DES Encryption Algorithm

INPUT: Binary message block $M=m_1,...,m_{64}$ and the binary secret key $K=k_1,...,k_{64}$, where the eight parity bits are $k_8,k_{16},...,k_{64}$.
OUTPUT: The ciphertext block $C=c_1,...,c_{64}$.

1. Call DES Key Schedule Algorithm with K as an input to obtain sixteen 48-bit subkeys K_i, $1 \le i \le 16$
2. Permute the 64-bit vector M using IP, and split the result equally into left half L_0 and right half R_0 (both are binary vectors of length 32)
3. For i from 1 to 16 do the following:
   a) $L_i=R_{i-1}$

b) $R_i=L_{i-1} \oplus P(S(E(R_{i-1}) \oplus K_i))$

4. $C=IP^{-1}(R_{16} \| L_{16})$, where $\|$ denotes concatenation of two binary strings.

DES Key Schedule Algorithm

INPUT: Binary string key $K=k_1,\ldots,k_{64}$.
OUTPUT: Sixteen 48-bit subkeys $K_1,\ldots,K_{16}$.

1. Obtain the 56-bit vector $PC_1(K)$, and split it equally into left half C_0 and right half D_0 (both are binary vectors of length 28).
2. For i from 1 to 16 do the following:
   a) If $i \in \{1,2,9,16\}$ then do the following:
      - Compute C_i by cyclically shifting C_{i-1} one place to the left.
      - Compute D_i by cyclically shifting D_{i-1} one place to the left.

   b) If $i \notin \{1,2,9,16\}$ then do the following:
      - Compute C_i by cyclically shifting C_{i-1} two places to the left.
      - Compute D_i by cyclically shifting D_{i-1} two places to the left.

   c) $K_i=PC_2(C_i \| D_i)$, where $\|$ denotes concatenation of two binary strings.

A nice feature of DES is that the decryption is performed by using the same algorithms, except that the subkeys are reversed. That is, after the key schedule call, K_i and K_{17-i} $(1 \le i \le 8)$ are swapped during the decryption process. Both encryption and decryption procedures are suitable for hardware implementation, which is another nice feature of DES.

4.1.1 An Example of DES with Test Vectors

In this example, the 16×16 tiny binary image "Lena" is evenly partitioned into four 8×8 quadrants, which correspond to NW, NE, SW, and SE corners of the image. The pixel values of each quadrant are taken row by row from top to bottom to form a 64-bit binary vector (black pixel is represented as 0 and white pixel as 1). Each of

the four 64-bit vectors is then taken as the input to the DES Encryption Algorithm along with the 64-bit secret key K, which is in this example taken to be "1999DE5FADE52AE5" in hexadecimal. The encrypted vectors are used to create four 8×8 pixel quadrants that are put together to form an encrypted 16×16 image as shown in Figure 4.2-b. After the key schedule procedure, the following sixteen 48-bit subkeys are created:

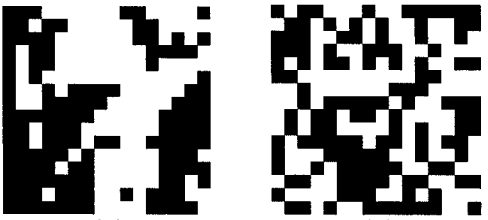
I	K_i	K_{i+1}
1	475663CDF64F	6C8339799F69
3	87903B5AFC3E	AF0AD26D7DBC
5	3E7A88A978FB	1A3558E7DA37
7	4C4C5D970FFE	47E90C9D9BD5
9	D605ADFD22EC	DB8205F0FACF
11	099AAE36B6BF	B030EEBF3DE3
13	B04E602EEB73	C07B3477ED56
15	84B755ED85DA	EA4516D7B7F8

When encrypting the first quadrant (NW) that corresponds to the 64-bit value "1F270F4F4F5F6860", the DES Encryption Algorithm generates the following values for L_i and R_i, $0 \leq i \leq 16$:

i	L_i	R_i	L_{i+1}	R_{i+1}
0	F8213F3F	00C27D3F	00C27D3F	F44AA20D
2	F44AA20D	83204D9E	83204D9E	7C057AA7
4	7C057AA7	56D6CCE4	56D6CCE4	0119B900
6	0119B900	D4CBA97F	D4CBA97F	12792418
8	12792418	FBF93ED7	FBF93ED7	EBC6AA7F
10	EBC6AA7F	8DD919D2	8DD919D2	F8E67873
12	F8E67873	91A5AFDB	91A5AFDB	63D39EF5
14	63D39EF5	44DCD7E7	44DCD7E7	C2206579
16	C2206579	D1FE6B66	--	--

Finally, the algorithm produces the ciphertext "4E951916523FDFD0", by permuting the 64-bit vector $R_{16} \parallel L_{16}$ according to IP^{-1}.

Quadrant	Plaintext	Ciphertext
NW	1F270F4F4F5F6860	4E951916523FDFD0
NE	9ECFCAE1EFFE7CF8	C06C6D37ACE78F58
SW	0123220509011101	A4A7417068987896
SE	F0F060F1F0F2A3E1	BCAC296D334CD706



(a) (b)

Figure 4.2 Encrypting the 16×16 binary image "Lena" using DES with key "1999DE5FADE52AE5": (a) the original image, (b) the encrypted image.

4.2 INTERNATIONAL DATA ENCRYPTION ALGORITHM (IDEA)

International Data Encryption Algorithm (IDEA) was patented by Massey and Lei in May 23, 1993 with a patent number 5,214,703 [10]. This was the commercialization of the authors' previous work, and it quickly became the cryptosystem of choice for many security applications. Along with DES and AES, IDEA is one of the most widely used symmetric-key cryptosystems. For example, IDEA was used as an integral part of the popular secure email software application *Pretty Good Privacy* (PGP). In the previous versions of PGP, IDEA was used as the symmetric-key cryptosystem in hybrid combination with RSA public-key cryptosystem. Later versions of PGP have IDEA as one of the choices for the symmetric-key cryptosystem that user can select.

IDEA encrypts a 64-bit plaintext block into a 64-bit ciphertext block. In doing so, unlike DES, IDEA uses a 128-bit secret key. This is quite an improvement with respect to the security level since DES essentially relies on a 56-bit secret key. Like most symmetric block ciphers, IDEA consists of the encryption/decryption part, here

denoted by the *IDEA Encryption Algorithm*, and the key schedule part, here denoted by the *IDEA Key Schedule Algorithm*. The IDEA Encryption Algorithm executes 8 rounds that involve a special (modified) multiplication of two 16-bit unsigned integers modulo $2^{16} + 1$, denoted by the symbol $\otimes$. Namely, this modified multiplication works the same way as the ordinary modular multiplication, except that the operands with zero values are replaced by the value 2^{16} prior the multiplication, and if the result of the multiplication is 2^{16} it is replaced by 0. The details of the IDEA Encryption Algorithm and the IDEA Key Schedule Algorithm are presented next.

IDEA Encryption Algorithm

INPUT: A binary message block $M=m_1,\ldots,m_{64}$ and the binary secret key $K=k_1,\ldots,k_{128}$.
OUTPUT: The ciphertext block $C=c_1,\ldots,c_{64}$.

1. Call the IDEA Key Schedule Algorithm with K as an input to obtain 52 binary subkeys of size 16, denoted by $K_1^{(i)},\ldots,K_6^{(i)}$, $1 \le i \le 8$, and $K_1^{(9)},\ldots,K_4^{(9)}$.
2. Partition M evenly into four 16-bit parts X_1, X_2, X_3, and X_4, so that $X_1 \,||\, X_2 \,||\, X_3 \,||\, X_4 = M$.
3. For i from 1 to 8 do the following:
   a) $X_1 = X_1 \otimes K_1^{(i)} \pmod{2^{16}+1}$
   b) $X_2 = X_2 + K_2^{(i)} \pmod{2^{16}}$
   c) $X_3 = X_3 + K_3^{(i)} \pmod{2^{16}}$
   d) $X_4 = X_4 \otimes K_4^{(i)} \pmod{2^{16}+1}$
   e) $T_1 = K_5^{(i)} \otimes (X_1 \oplus X_3) \pmod{2^{16}+1}$
   f) $T_2 = T_1 + (X_2 \oplus X_4) \pmod{2^{16}}$
   g) $T_2 = K_6^{(i)} \otimes T_2 \pmod{2^{16}+1}$
   h) $T_3 = T_1 + T_2 \pmod{2^{16}}$
   i) $X_1 = X_1 \oplus T_2$
   j) $T_4 = X_2 \oplus T_3$
   k) $X_2 = X_3 \oplus T_2$
   l) $X_3 = T_4$
   m) $X_4 = X_4 \oplus T_3$
4. Compute T_i, $1 \le i \le 4$, as follows:
   a) $T_1 = X_1 \otimes K_1^{(9)} \pmod{2^{16}+1}$
   b) $T_2 = X_3 + K_2^{(9)} \pmod{2^{16}}$
   c) $T_3 = X_2 + K_3^{(9)} \pmod{2^{16}}$
   d) $T_4 = X_4 \otimes K_4^{(9)} \pmod{2^{16}+1}$

Output $C=T_1 \| T_2 \| T_3 \| T_4$.

IDEA Key Schedule Algorithm

INPUT: A 128-bit secret key $K=k_1,\ldots,k_{128}$.
OUTPUT: 52 binary subkeys of size 16, denoted by $K_1^{(i)},\ldots,K_6^{(i)}$, $1 \le i \le 8$, and $K_1^{(9)},\ldots,K_4^{(9)}$.

1. Order the 52 subkeys in the following way: $K_1^{(i)},\ldots,K_6^{(i)}$, $1 \le i \le 8$, and $K_1^{(9)},\ldots,K_4^{(9)}$. (For example, $K_1^{(1)}$ is the first subkey, $K_6^{(1)}$ is the sixth subkey, $K_1^{(2)}$ is the seventh subkey, and $K_4^{(9)}$ is the fifty second subkey)
2. For i from 1 to 6 do the following:
   a) Partition K into eight 16-bit blocks $B_1,\ldots,B_8$, so that $B_1 \| \ldots \| B_8 = K$
   b) For j from 1 to 8 do the following:
      - $N = (i-1)\times 8 + j$
      - Set the N^{th} subkey to B_j
   c) Perform a left cyclic shift on K by 25 places
3. Partition the first half of K into four 16-bit blocks B_1, B_2, B_3, and B_4, so that $B_1 \| B_2 \| B_3 \| B_4$ constitutes the first half of K.
4. Compute the last four subkeys as follows: $K_i^{(9)} = B_i$, $1 \le i \le 4$.

The decryption algorithm is the same as the IDEA Encryption Algorithm except for the following changes: the input message M is replaced by a valid ciphertext C, and although the secret key K remains the same, it undergoes a different key schedule procedure. The modified key schedule algorithm for decryption is presented next.

IDEA Key Schedule Algorithm (decryption-only)

INPUT: A 128-bit secret key $K=k_1,\ldots,k_{128}$.
OUTPUT: 52 binary subkeys of size 16, denoted by $K_1^{(i)},\ldots,K_6^{(i)}$, $1 \le i \le 8$, and $K_1^{(9)},\ldots,K_4^{(9)}$.

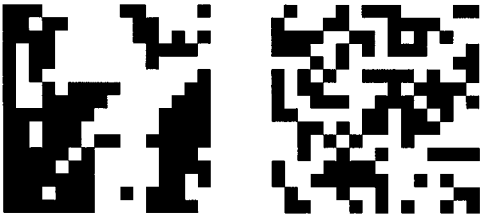
1. Obtain 52 subkeys $K_1^{(i)}, \ldots, K_6^{(i)}$, $1 \le i \le 8$, and $K_1^{(9)}, \ldots, K_4^{(9)}$ using the IDEA Key Schedule Algorithm for encryption.
2. For i from 1 to 8 do the following:

a) Set $L_j^{(i)}$ to the multiplicative inverse of $K_j^{(10-i)}$ modulo $(2^{16}+1)$, $j \in \{1,4\}$.
b) If i is 1 then do the following:
   - Set $L_j^{(i)}$ to the additive inverse of $K_j^{(10-i)}$ modulo 2^{16}, $j \in \{2,3\}$.
   - Set $L_j^{(i)}$ to $K_j^{(9-i)}$, $j \in \{5,6\}$.
c) If $2 \leq i \leq 8$ then do the following:
   - Set $L_2^{(i)}$ to the additive inverse of $K_3^{(10-i)}$ modulo 2^{16}, and set $L_3^{(i)}$ to the additive inverse of $K_2^{(10-i)}$ modulo 2^{16}.
   - Set $L_j^{(i)}$ to $K_j^{(9-i)}$, $j \in \{5,6\}$.
d) If i is 9 then do the following:
   - Set $L_j^{(i)}$ to the additive inverse of $K_j^{(10-i)}$ modulo 2^{16}, $j \in \{2,3\}$.

Output L's as the subkeys

4.2.1 An Example of IDEA with Test Vectors

To encrypt the 16×16 binary image "Lena", we evenly partition it into four 8×8 quadrants corresponding to the NW, NE, SW, and SE corners of the image. The pixel values form a 64-bit binary vector (black pixel is represented as 0 and white pixel as 1). The four 64-bit vectors are then taken as the input to the IDEA Encryption Algorithm along with the 126-bit secret key K, which is in this example taken to be "111976DE5052319931DEA05262002AE5" in hexadecimal. The encrypted vectors are put together to form an encrypted 16×16 image shown in Figure 4.3-b.



(a) (b)

Figure 4.3 Encrypting the 16×16 binary image "Lena" using IDEA with key "111976DE5052319931DEA05262002AE5": (a) the original image, (b) the encrypted image.

Quadrant	Plaintext	Ciphertext
NW	1F270F4F4F5F6860	BA73EB560C1A4F63
NE	9ECFCAE1EFFE7CF8	3C87FC16AB8EA112
SW	0123220509011101	3ECA7060D5DBBB9D
SE	F0F060F1F0F2A3E1	4B47FF6BDF707C5A

The following two tables of test vectors show the key expansion values and an encryption of the first block (the NW corner of "Lena"):

Round i	$K_1^{(i)}$	$K_2^{(i)}$	$K_3^{(i)}$	$K_4^{(i)}$	$K_5^{(i)}$	$K_6^{(i)}$
1	1119	76DE	5052	3199	31DE	A052
2	6200	2AE5	BCA0	A463	3263	BD40
3	A4C4	0055	CA22	32ED	C664	C77A
4	8149	8800	AB94	4465	DB79	4148
5	F502	9310	0157	2888	CBB6	F282
6	918C	C98E	2002	AE51	1197	6DE5
7	0523	1993	1DEA	526	A223	2EDB
8	CA0A	4633	263B	D40A	4C40	055C
9	B794	148C	664C	77A8	--	--

Round	X_1	X_2	X_3	X_4
1	F072	F078	E459	CF5C
2	3DB3	DABF	E52C	651F
3	CF94	A18D	CA21	86A3
4	94F1	BD24	88EF	E735
5	9B69	5178	0DBE	AA04
6	B19F	824E	0640	12C6
7	0D20	4927	670C	5BE9
8	B0A8	850A	F78E	462E
9	BA73	EB56	0C1A	4F63

4.3 ADVANCED ENCRYPTION STANDARD (AES)

Advanced Encryption Standard, or shortly AES, is essentially a version of Rijndael [29] symmetric-key cryptosystem that processes 128-bit data blocks using cipher keys with lengths of either 128, 192, or 256 bits. Although Rijndael cryptosystem is more scalable and can handle different key sizes and data blocksizes, AES version

of Rijndael [11] is limited to the parameters shown in Table 4.1. Note that the blocksize is always fixed to 128-bits, while there are three choices for the key length and the number of rounds. The highest security mode is AES-256, which takes a 256-bit key and performs 14 rounds during encryption and decryption. However, this is the slowest mode and for faster applications one should use AES-128, since its security level is also extremely good by today's standards. Be aware that AES-128 uses a secret key that is more than two times larger than the effective key used in DES algorithm.

Table 4.1 The basic AES parameters.

AES Version	N_k (Key-length in words)	N_b (Blocksize in words)	N_r (Number of rounds)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Just like its predecessor algorithms DES and IDEA, AES algorithm contains a key schedule part. However, before key schedule takes place during the encryption and decryption stages, the key is expanded using the *key expansion algorithm*. The key expansion algorithm takes the (N_k×4)-byte key K, and expands it into an N_b×(N_r+1) array of words (32-bit vectors) W. The pseudo-code for AES key expansion algorithm is presented next.

AES Key Expansion Algorithm

INPUT: An N_k×4-byte key K, and integers N_k, N_r, and N_b.
OUTPUT: An array of N_b×(N_r+1) words W.

1. Set i to 0
2. While i<N_k do the following:
   a. $W[i] = K[4\times i] \,||\, K[4\times i+1] \,||\, K[4\times i+2] \,||\, K[4\times i+3]$
   b. $i = i + 1$
3. Set i to N_k
4. While i<N_b×(N_r+1) do the following:
   a. $T = W[i-1]$
   b. If i mod N_k is 0, then $T = SubWord(RotWord(T)) \oplus Rcon[\lfloor i/N_k \rfloor]$

c. Otherwise, if $N_k>6$ and i (mod N_k) is 4, then T = *SubWord*(T)
d. $W[i] = W[i-N_k] \oplus T$
e. $i = i + 1$

5. Output W

In this algorithm, *SubWord* is a function that takes a four-byte input word and applies the S-box to each of the four bytes to produce an output word. The function *RotWord* takes a word $a_0 \| a_1 \| a_2 \| a_3$ as input, performs a cyclic permutation, and returns the word $a_1 \| a_2 \| a_3 \| a_0$. The round constant word array, *Rcon*[i], contains the values given by $x^{i-1} \| \{00\} \| \{00\} \| \{00\}$, with x^{i-1} being powers of x (x is denoted as {02}) in the field $GF(2^8)$. AES algorithm uses the following S-box:

Table 4.2. AES Encryption S-box: substitution values for the byte xy (in hexadecimal representation).

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

The AES algorithm's internal operations are performed on a two-dimensional array of bytes called the *state*. The state consists of 4 rows of bytes, each containing N_b bytes. At the beginning of encryption and decryption procedures, the input array *IN*, is copied to the state array S according to the following rule: $S[r, c] = IN[r + 4\times c]$, where $0 \le r < 4$ and $0 \le c < N_b$. Similarly, at the end of each

procedure, S is copied to the output array *OUT* as follows: $OUT[r + 4\times c] = S[r, c]$, where $0 \le r < 4$ and $0 \le c < N_b$. In this notation, r denotes a row and c denotes a column. AES makes use of the following three transformations: *SubBytes*, *ShiftRows* and *MixColumns*. The *SubBytes* transformation is essentially an AES S-box, which is a non-linear byte substitution transformation that operates independently on each byte of the state. The simplest representation of the S-box function is by the lookup table given in Table 4.2. In the *ShiftRows* transformation, the bytes in the last three rows of the state are cyclically shifted over different numbers of bytes, while the first row is not shifted. The *ShiftRows* transformation is defined as follows: $s'[r, c] = s[r, (c + shift(r, Nb)) \bmod Nb]$ for $0 \le r < 4$ and $0 \le c < N_b$, where the shift value $shift(r, Nb)$ is defined in the following way: $shift(1, 4) = 1$, $shift(2, 4) = 2$, and $shift(3, 4) = 3$. The *MixColumns* transformation operates on the state column-by-column. The columns are considered as polynomials over the Galois Field $GF(2^8)$ and multiplied modulo $x^4 + 1$ with a fixed polynomial $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$. The notation $\{h_1h_2\}$ denotes a byte in the hexadecimal format. This can be accomplished by applying the following four equations to the state columns:

1. $S'[0, c] = (\{02\} \bullet S[0, c]) \oplus (\{03\} \bullet S[1, c]) \oplus S[2, c] \oplus S[3, c]$
2. $S'[1, c] = S[0, c] \oplus (\{02\} \bullet S[1, c]) \oplus (\{03\} \bullet S[2, c]) \oplus S[3, c]$
3. $S'[2, c] = S[0, c] \oplus S[1, c] \oplus (\{02\} \bullet S[2, c]) \oplus (\{03\} \bullet S[3, c])$
4. $S'[3, c] = (\{03\} \bullet S[0, c]) \oplus S[1, c] \oplus S[2, c] \oplus (\{02\} \bullet S[3, c])$

Here, the $\bullet$ operation denotes multiplication in $GF(2^8)$ modulo the polynomial $x^4 + 1$, while the $\oplus$, as usual, denotes the bitwise XOR operation. Finally, AES algorithm uses a function *AddRoundKey* that refreshes the state vector S with new values depending on W and the round index R. The function transforms S into S' in the following way:

$$S'[0,c] \,||\, S'[1,c] \,||\, S'[2,c] \,||\, S'[3,c] = S[0,c] \,||\, S[1,c] \,||\, S[2,c] \,||\, S[3,c] \oplus W[R],$$

where $||$ denotes the bitwise concatenation, and $0 \le c < N_b$. The following pseudo-code realizes the AES encryption with an input parameter W that was previously obtained by using the key expansion algorithm.

AES Encryption Algorithm

INPUT: A plaintext block *IN* consisting of $4 \times N_b$ bytes, a key expansion *W* consisting of $N_b \times (N_r+1)$ words, and integers N_b and N_r.
OUTPUT: A ciphertext block *OUT* consisting of $4 \times N_b$ bytes.

1. Copy array *IN* into the state array *S* using the following rule: $S[r, c] = IN[r+4 \times c]$, where $0 \le r < 4$ and $0 \le c < N_b$
2. Perform *AddRoundKey*(*S*,*W*,0)
3. For *R* from 1 to $N_r - 1$ do the following:
   a. Perform *SubBytes*(*S*)
   b. Perform *ShiftRows* (*S*)
   c. Perform *MixColumns*(*S*)
   d. Perform *AddRoundKey*(*S*,*W*,*R*)
4. Perform *SubBytes*(*S*)
5. Perform *ShiftRows*(*S*)
6. Perform *AddRoundKey*(*S*,*W*,N_r)
7. Copy the state array *S* into the output array *OUT* using the following rule: $OUT[r+4 \times c] = S[r, c]$, where $0 \le r < 4$ and $0 \le c < N_b$
8. Output *OUT*

The above algorithm creates a 128-bit output that represents an AES encrypted block. In order to decrypt such block, we run *AES Decryption Algorithm*. As a preprocessing step for AES decryption, we need to obtain the identical vector *W* that was used during the encryption. To do so, we have to know the secret key *K* used in the encryption process. If *K* is known, *W* can be easily obtained by using the key expansion algorithm.

The AES decryption process uses the following three transformations: *InvShiftRows*, *InvSubBytes*, and *InvMixColums*. As the name indicates, these are essentially inverse transformations of *ShiftRows*, *SubBytes*, and *MixColumns* used during the encryption process. The *InvShiftRows* is the inverse of the *ShiftRows* transformation, and it is defined as follows: $s[r, c] = s'[r, (c + shift(r, N_b)) \bmod N_b]$ for $0 \le r < 4$ and $0 \le c < N_b$. Similarly, *InvSubBytes* is the inverse of the byte substitution transformation, in which the

inverse S-box is applied to each byte of the state. The inverse S-box used in the *InvSubBytes* transformation is presented in Table 4.3.

The *InvMixColumns* transformation operates on the state column-by-column, treating each column as a polynomial over the field GF(2^8) modulo $x^4 + 1$ with a fixed polynomial $a^{-1}(x)$, given by $a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$. This can be accomplished by applying the following four equations to the state columns:

1. $s'[0, c] = (\{0e\} \bullet s[0, c]) \otimes (\{0b\} \bullet s[1, c]) \otimes (\{0d\} \bullet s[2, c]) \otimes (\{09\} \bullet s[3, c])$
2. $s'[1, c] = (\{09\} \bullet s[0, c]) \otimes (\{0e\} \bullet s[1, c]) \otimes (\{0b\} \bullet s[2, c]) \otimes (\{0d\} \bullet s[3, c])$
3. $s'[2, c] = (\{0d\} \bullet s[0, c]) \otimes (\{09\} \bullet s[1, c]) \otimes (\{0e\} \bullet s[2, c]) \otimes (\{0b\} \bullet s[3, c])$
4. $s'[3, c] = (\{0b\} \bullet s[0, c]) \otimes (\{0d\} \bullet s[1, c]) \otimes (\{09\} \bullet s[2, c]) \otimes (\{0e\} \bullet s[3, c])$

Table 4.3. AES Decryption S-box: substitution values for the byte xy (in hexadecimal representation).

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Finally, the AES decryption process uses the *AddRoundKey* function that is identical to the one from the AES encryption procedure, since this transformation is its own inverse. The pseudo-code for AES Decryption Algorithm is presented next.

AES Decryption Algorithm

INPUT: A ciphertext block *IN* consisting of $4\times N_b$ bytes, a key expansion *W* consisting of $N_b\times(N_r+1)$ words, and integers N_b and N_r.

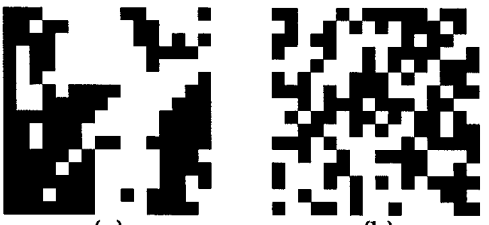
OUTPUT: A decrypted plaintext block *OUT* consisting of $4\times N_b$ bytes.

1. Copy array *IN* into the state array *S* using the following rule: $S[r, c] = IN[r+4\times c]$, where $0\le r<4$ and $0\le c<N_b$
2. Perform *AddRoundKey*(S,W,N_r)
3. For *R* from N_r-1 down-to 1 do the following:
   a. Perform *InvShiftRows*(S)
   b. Perform *InvSubBytes*(S)
   c. Perform *AddRoundKey*(S,W,R)
   d. Perform *InvMixColumns*(S)
4. Perform *InvShiftRows*(S)
5. Perform *InvSubBytes*(S)
6. Perform *AddRoundKey*$(S,W,0)$
7. Copy the state array *S* into the output array *OUT* using the following rule: $OUT[r+4\times c] = S[r, c]$, where $0\le r<4$ and $0\le c<N_b$
8. Output *OUT*

The security level of AES is to this day substantial since there are no known effective attacks discovered thus far. For more information about AES, refer to [38].

4.3.1 An Example of AES-128 with Test Vectors

In this example, the 16×16 binary image "Lena" is partitioned into two 8×16 halves, which correspond to the top half and the bottom half the image. The pixel values of each quadrant are taken row by row from top to bottom to form two 128-bit binary vectors, which are then taken as the input to the AES Encryption Algorithm along with the secret 128-bit key *K*, which is in this example taken to be "111976DE5052319931DEA05262002AE5" in hexadecimal.



(a) (b)

Figure 4.4. Encrypting the tiny 16×16 binary image "Lena" using AES-128 with key "111976DE5052319931DEA05262002AE5": (a) the original image, (b) the encrypted image.

The encrypted vectors are used to create two 8×16 top and bottom pixel segments that are put together to form an encrypted 16×16 image as shown in Figure 4.4-b.

Plaintext (Top)	1F9E27CF0FCA4FE14FEF5FFE687C60F8
Ciphertext (Top)	3A09B54527D37686EEA85C591474D113

Plaintext (Bottom)	01F023F0226005F109F001F211A301E1
Ciphertext (Bottom)	A4A5CD6417D0DB0E73CD7550DDF965BC

The following tables of test vectors show the key expansion values and an encryption of the first block (the top half of "Lena"):

I	*W[I]*	*W[I+1]*	*W[i+2]*	*W[i+3]*
0	111976DE	50523199	31DEA052	62002AE5
4	73FCAF74	23AE9EED	12703EBF	7070145A
8	20061125	03A88FC8	11D8B177	61A8A52D
12	E600C9CA	E5A84602	F470F775	95D85258
16	8F00A3E0	6AA8E5E2	9ED81297	0B0040CF
20	FC0929CB	96A1CC29	0879DEBE	3.80E+74
24	6A028AB0	FCA34699	F4DA9827	F7A30656
28	206D3BD8	DCCE7D41	2814E566	DFB7E330
32	097C3F46	D5B24207	FDA6A761	22114451
36	9067EED5	45D5ACD2	B8730BB3	9A624FE2
40	0CE3766D	4936DABF	F145D10C	6B279EEE

Round	The State Vector
1	35B936E5 83B92244 A00785AC 95D62574
2	64B53696 99C21810 3F85BA2E 03D7FA81
3	8AF3E73D 1D07C68A A2A0C38F BB388A8D
4	30891C97 1B6A4B02 B4FF658B 954ACAC9
5	E5296D16 995797BD 59AB1B33 9CF5C07D
6	2D5FD864 B12E2BEE 2C16E553 0BB8F942
7	B9FA0DB6 5E075CF3 DDD1CAFD 01FE2984
8	23FD95E0 3C47E7C2 A8A88E61 501E487A
9	B4375A8E 006E3E3F 73593892 F131E50C
10	3A09B545 27D37686 EEA85C59 1474D113

4.4 RSA PUBLIC-KEY CRYPTOSYSTEM

The RSA cryptosystem is the most widely used public-key cryptosystem today. It was invented by Ronald L. Rivest, Adi Shamir and Leonard Adleman in 1978 [17], and later patented [18]. The name RSA comes from the authors' last name initials. The RSA scheme relies on the difficulty of factoring large integers. In essence, the scheme is quite simple.

RSA Key Generation Algorithm

INPUT: An integer N, indicating an N-bit RSA strength.
OUTPUT: Integers d, e, and n.

1. Generate two pseudo-random primes[3] p and q of roughly the same size so that $p \times q$ is smaller than but close to 2^N
2. Set n to $p \times q$ and compute the Euler Phi function of n, $\Phi(n)=(p-1)\times(q-1)$
3. Randomly select an integer e such that $1<e<\Phi(n)$ and GCD(e, $\Phi(n)$)=1

[3] There are efficient algorithms for generating pseudo-random primes (for example Miller-Rabin algorithm [30]). In addition, there are well-known weak selections for p and q (see [2]).

4. Compute an integer d to be the multiplicative inverse of e modulo $\Phi(n)$; i.e., $ed \equiv 1 \pmod{\Phi(n)}$ and $1<d<\Phi(n)$.

Output d, e, and n

A communicating party A uses the above algorithm to generate three integers: d_A, e_A, and n_A. The integers e_A and n_A are published as A's public key, while the integer d_A is kept secret as A's private-key. Since there is no efficient integer factorization algorithm (up to date), The RSA public information (n_A, e_A) alone cannot be used to compute the secret integer d_A.

Suppose Bob wants to confidentially send a message M to Alice using RSA. First, Bob has to encode message M into a sequence of integers where each entry is an integer inclusively between 0 and $n_{Alice}-1$. Then, each integer from this sequence undergoes the following transformation (the RSA Encryption Algorithm).

RSA Encryption Algorithm

INPUT: A plaintext integer m satisfying $0 \le m < n$, and recipient's public key pair n and e.

OUTPUT: An integer m', which is an RSA encryption of m.

1. Set m' to $m^e \pmod{n}$
2. Output m'

Since the receiver of message M is Alice, Bob has to use the integers n_{Alice} and e_{Alice} in the encryption procedure, which he can easily obtain since Alice had previously published her public key pair (n_{Alice}, e_{Alice}). Once the sequence of encrypted integers is obtained, Bob sends it to Alice.

On the other end, Alice receives the sequence of encrypted integers, and for each encrypted integer, she computes the original integer using the RSA Decryption Algorithm. Finally, she decodes the sequence of decrypted integers to recover message M.

RSA Decryption Algorithm

INPUT: A ciphertext integer m' satisfying $0 \leq m' < n$, and own private-key pair n and d.
OUTPUT: An integer m, which is an RSA decryption of m'.

1. Set m to $(m')^d \pmod{n}$
2. Output m

Note that only Alice is able to recover message M since only Alice knows the secret key d_{Alice}. The only obvious way for an adversary to recover d_{Alice} is to factor n_{Alice} and generate the unknown value d_{Alice}. However, this is computationally difficult.

4.4.1 An Example of RSA

In today's world, the typical strength of RSA should be at least 1024 bits (2048 bits is more recommended). However, for the simplicity of argument, in the example that follows we chose N to be only 20.

Suppose both Alice and Bob are part of an RSA communication group. When Alice joined the group, her public and private keys were generated using the RSA Key Generation Algorithm with $N = 20$. The algorithm randomly selected p_{Alice} to be 653 and q_{Alice} to be 719, and thus $n_{Alice} = p_{Alice} \times q_{Alice} = 469507$ and $\Phi(n_{Alice}) = (p_{Alice}-1) \times (q_{Alice}-1) = 468136$. Next, a public exponent e_{Alice} was selected to be 3, which was possible since GCD(3,468136) = 1. From there, the private key integer d_{Alice}, was calculated as follows:

$$d_{Alice} = e_{Alice}^{-1} \pmod{\Phi(n_{Alice})} = 3^{-1} \pmod{468136} = 312091.$$

It is worth mentioning that finding a multiplicative inverse in the above equation is usually accomplished by applying an extended Euclidean algorithm. Furthermore, all of the modular operations involved in RSA can be very efficiently implemented even for extremely large integers, which are used in commercial applications of RSA. Still, with such large-scale computations, RSA and other public-key cryptosystems run much slower then most modern symmetric-key cryptosystems.

Lets say that Bob wishes to confidentially send a message M = "LENA" to Alice. First, Bob encodes this message into an integer between 0 and n_{Alice}-1. The encoding is used by the members of the communication group is as follows. If c is a letter of the English alphabet, let $ind(c)$ denote its zero-based index within the alphabet. The message is divided into four-letter blocks and each block $c_0c_1c_2c_3$ is encoded into the following integer value:

$$\sum_{i=0}^{3} ind(c_i) \times 26^i$$

Therefore, Bob encodes "LENA" into $11\times1 + 4\times26 + 13\times26^2 + 0\times26^3 = 8903$. Then, Bob uses Alice's public key e_{Alice} and n_{Alice} to obtain the following encryption of M:

$$8903^3 \pmod{469507} = 432638.$$

When Alice receives the encrypted message 432638, she decrypts it using her private key d_{Alice} as follows:

$$432638^{312091} \pmod{469507} = 8903.$$

Finally, she easily decodes the message 8903 back to "LENA".

4.5 ELGAMAL PUBLIC-KEY CRYPTOSYSTEM

Another popular public-key cryptosystem is the ElGamal cryptosystem. The ElGamal scheme was created in 1984 by Taher ElGamal [19], and it is based on the intractability of the discrete logarithm problem. The scheme is heavily motivated by the Diffie-Hellman protocol [13] (see next section). The following pseudo-code represents the ElGamal key generation procedure.

ElGamal Key Generation Algorithm

INPUT: An integer N, indicating an N-bit ElGamal strength.
OUTPUT: Integers p, g, a and h.

1. Select a pseudo-random number p that is close to but smaller than 2^N

2. Select a pseudo-random integer g which must also be a generator of the multiplicative group of integers modulo p
3. Select a pseudo-random integer a, such that $0<a<p-1$
4. Compute $h = g^a \bmod p$
5. Output p, g, a, and h

A communicating party A uses the above algorithm to generate four integers: p_A, g_A, a_A and h_A. The integers p_A, g_A and h_A are published as A's public key, while the integer a_A is kept secret as A's private-key. Since there is no efficient algorithms for solving the discrete logarithm problem (up to date), ElGamal public information (p_A, g_A, h_A) alone cannot be used to compute the secret integer a_A.

If Bob wants to secretly send a message M to Alice using ElGamal, he has to encode message M into a sequence of integers where each entry is an integer inclusively between 0 and $p_{Alice}-1$. Each integer from this sequence undergoes the ElGamal Encryption Algorithm.

ElGamal Encryption Algorithm

INPUT: A plaintext integer m satisfying $0 \leq m < p$, and recipient's public key p, g, and h.
OUTPUT: A pair of integers (v, w) (the encryption of m).

1. Select a pseudo-random integer k, such that $0<k<p-1$
2. Compute $v = g^k \bmod p$
3. Compute $w = m \times h^k \bmod p$
4. Output (v, w)

Since the receiver of message M is Alice, Bob has to use the publicly known integers n_{Alice} and e_{Alice} as the input to the encryption procedure. Once the sequence of encrypted integers is obtained, Bob sends it to Alice.

When Alice receives the sequence of encrypted integers, she computes the original integers using the ElGamal Decryption Algorithm with each encrypted integer. At the end, she decodes the sequence of decrypted integers to recover message M.

ElGamal Decryption Algorithm

INPUT: A ciphertext integers v and w, and own private-key a and p.
OUTPUT: An integer m, which is an ElGamal decryption of (v, w).

1. Compute $t = v^{p-1-a} \bmod p$
2. Compute $m = t \times w \bmod p$
3. Output m

Since Alice is the only one that knows the secret key a_{Alice}, only Alice is able to recover message M. An eavesdropper must solve a discrete logarithm problem in order to compute a_{Alice}, but as we already know, this is computationally difficult.

4.5.1 An Example of ElGamal

Suppose both Alice and Bob are part of an ElGamal communication group. Upon joining the group, Alice's public and private keys were generated using ElGamal Key Generation Algorithm with $N = 20$. The algorithm randomly selected p_{Alice} to be 1048573, a generator g_{Alice} to be 1033, and a_{Alice} to be 276138. Thus, h_{Alice} was set to 377346.

If Bob wishes to confidentially send a message M = "LENA" to Alice, he encodes this message into $11\times1 + 4\times26 + 13\times26^2 + 0\times26^3 = 8903$, by applying the encoding scheme described in the previous section. Then, Bob generates a random number $k = 783129$, and uses Alice's public key information p_{Alice}, g_{Alice}, and h_{Alice} to obtain the following encryption of M:

$$1033^{783129} \pmod{1048573} = 680615,$$
$$8903\times377346^{783129} \pmod{1048573} = 259686.$$

When Alice receives the encrypted message (680615, 259686), she decrypts it using her private key a_{Alice} as follows:

$$680615^{1048573-1-276138}\times259686 \pmod{1048573} = 8903.$$

Finally, Alice decodes the message 8903 back to "LENA".

4.6 DIFFIE-HELLMAN KEY EXCHANGE PROTOCOL

The first practical key exchange protocol was created by Whitfield Diffie and Martin Hellman in 1976 [13]. This protocol, often referred to as the Diffie-Hellman protocol, allows two communicating parties to safely agree upon a shared secret. The communicants do not need to have any prior contact nor posses previously shared information, and yet they are able to share the secret securely using an insecure channel. The security of Diffie-Hellman protocol relies on the difficulty of solving the discrete logarithm problem. The following pseudo-code realizes Diffie-Hellman protocol:

Diffie-Hellman Key Exchange Protocol

SETUP: Communicants A and B are able to exchange messages over a potentially insecure channel. A prime p and an integer g, which is a generator of the multiplicative group of integers modulo p, were previously published.
RESULT: An integer K, that is known only to A and B.

1. A chooses a random integer x, such that $0<x<p-1$
2. A computes $a = g^x \bmod p$ and sends it to B
3. B chooses a random integer y, such that $0<y<p-1$
4. B computes $b = g^y \bmod p$ and sends it to A
5. A computes $K = b^x \bmod p$
6. B computes $K = a^y \bmod p$

Once the communicants A and B generate the shared secret K, they can use it as a secret session key together with a symmetric-key cryptosystem, such as AES, to encrypt all messages that are exchanged during the communication session.

The attacker that listens to the channel can obtain p, g, $a = g^x \bmod p$, and $b = g^y \bmod p$. Unfortunately, the attacker needs $K = b^x \bmod p = a^y \bmod p = g^{xy} \bmod p$. In order to calculate K, the attacker first has to solve a discrete logarithm problem to recover either x or y. This, however, is computationally hard.

4.6.1 An Example of Diffie-Hellman Protocol

Suppose Alice and Bob would like to carry out a secure communication session, where all messages would be encrypted with some symmetric-key cryptosystem. However, they have never met, and they have no shared secret that could be used as a secret key in the chosen symmetric-key cryptosystem. Here, the Diffie-Hellman protocol can help to generate such a session key.

Assume that the public Diffie-Hellman values p = 1017473 and g = 7789 are known to both Alice and Bob. According to the protocol, Alice chooses random integer x = 415492, calculates $a = 7789^{415492} \pmod{1017473} = 898132$, and sends it to Bob. On the other end, Bob randomly selects integer y = 725193, calculates $b = 7789^{725193} \pmod{1017473} = 497587$, and sends it to Alice. Now, Alice computes $K = 497587^{415492} \pmod{1017473} = 707860$, and Bob computes $K = 898132^{725193} \pmod{1017473} = 707860$, which they can use as the session key.

SUMMARY

Symmetric-key cryptosystems DES, IDEA, and AES are some of the most commonly used conventional cryptosystems. The Advanced Encryption Standard (AES) is the latest recommendation for the commercial encryption algorithm, and it supports three different security levels. The most popular and widely used public-key cryptosystems are RSA and ElGamal. In addition, these systems are often used to utilize digital signature schemes. Finally, the classical Diffie-Hellman key exchange protocol still represents a milestone in majority of applications that require a session key establishment.

Chapter 5

IMAGE ENCRYPTION ALGORITHMS

When dealing with still images, the security is often achieved by using the naïve approach to completely encrypt the entire image. However, there are number of applications for which the naïve based encryption is not suitable. For example, a limited bandwidth and processing power in small mobile devices calls for different approaches. In this section, we introduce some of the most important techniques for image encryption based on encrypting only certain parts of the image in order to reduce the amount of computation (selective image encryption). In addition, we discuss important image encryption approaches based on fast chaotic maps that encrypt either the entire image or a part of it. All of the methods discussed in this chapter are designed either for the uncompressed images or for the specific image compression formats. The methods designed for JPEG encoded images are usually extensible to protect the MPEG encoded videos.

5.1 IMAGE ENCRYPTION BY VAN DROOGENBROECK AND BENEDETT

The work of Van Droogenbroeck and Benedett was concentrated on degradation rather than highly secure encryption. In [31], they suggested two selective degradation methods. The first method was designed for the uncompressed (raster) images, while the second method was designed for the JPEG compressed images. Although the authors essentially suggested a one-time pad for encrypting the selected information, any conventional cryptosystem would be suitable for encrypting the selected bits.

5.1.1 Van Droogenbroeck-Benedett Algorithm I

Van Droogenbroeck-Benedett Algorithm I degrades uncompressed 8-bit grayscale images. Such images contain exactly 8 bitplanes. Each pixel takes 8 bits and each bit is a part of a different bitplane.

The most significant bit belongs to the most significant bitplane (MSB), while the least significant bit belongs to the least significant bitplane (LSB). Figure 5.1 shows an example of 8 different bitplanes of grayscale "Lena".

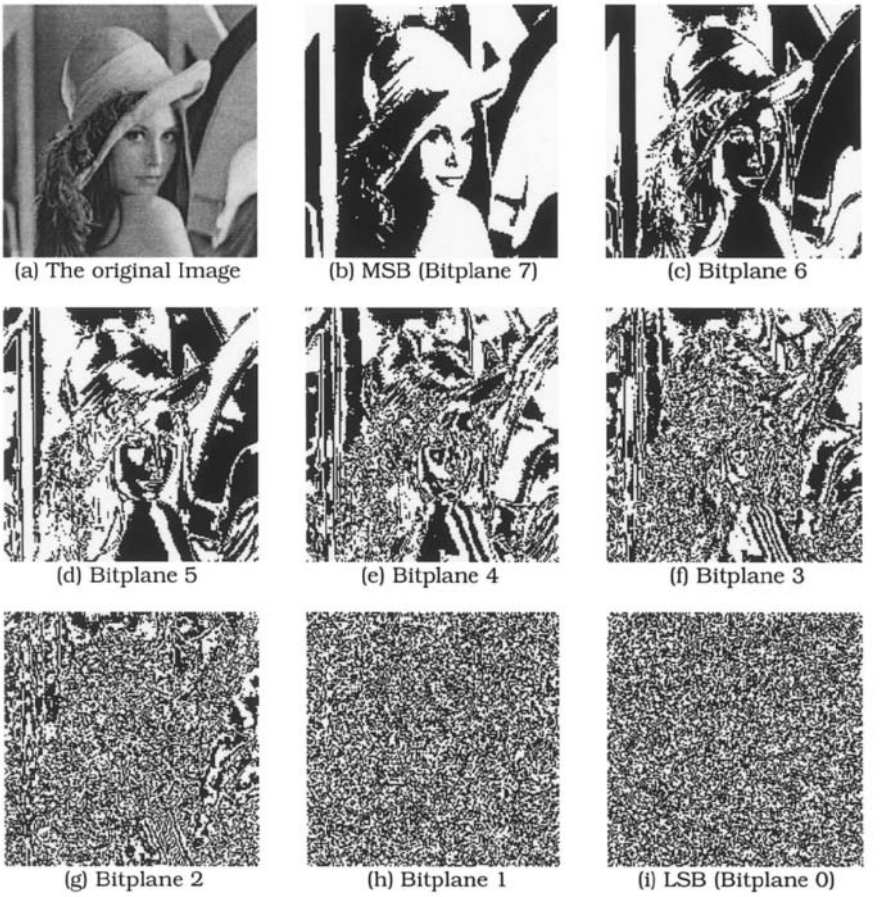


Figure 5.1. (a) The original grayscale "Lena", and (b)-(i) eight different bitplanes of "Lena".

Notice in Figure 5.1 that the most significant bitplanes show similarity and a high correlation with the original image, but the least significant bitplanes appear random and show no correlation with the original image. This observation was the basic idea behind the Van Droogenbroeck-Benedett Algorithm I. Instead of naively

encrypting the entire bitstream (i.e., all 8 bitplanes), the authors suggested encrypting at least 4-5 least significant bitplanes. This would result in visible quality degradation, but the image would still be recognizable. The least significant bitplanes appear random-like, and encrypting only the bitplanes that contain nearly uncorrelated values would decrease the vulnerability to the known-plaintext attacks, as well as the other statistical attacks.

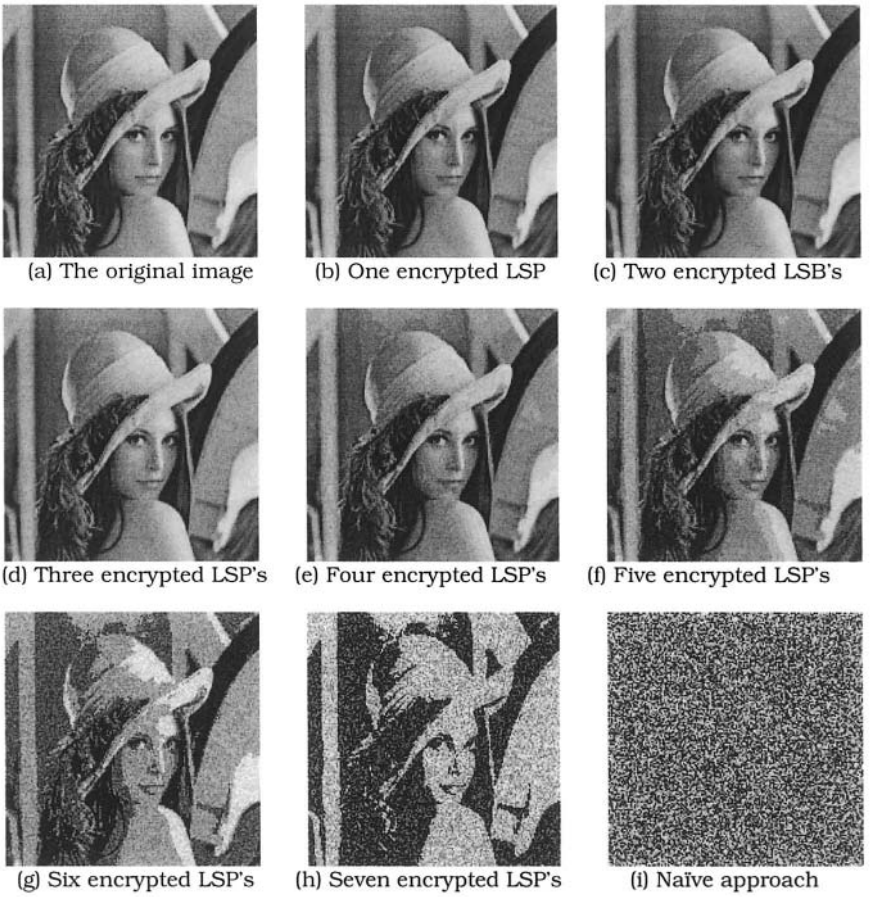


Figure 5.2. (a) The original grayscale "Lena", and (b)-(i) gradually increasing the number of least significant bitplanes (LSP's) that are encrypted. The image (i) has all bitplanes encrypted (the naïve approach).

The results from [31] showed that at least 4-5 bitplanes need to be encrypted before the image degradation is visually satisfactory. Figure 5.2 shows a visual degradation of "Lena" achieved by encrypting a different number of least significant bitplanes.

The algorithm is difficult to cryptanalyze from the point of view of recovering the original quality. As noted before, the encrypted information is highly uncorrelated, and as such, it would be difficult to perform statistical cryptanalysis. It is important to note that the quality degradation of the image is not sufficient for applications where high security is necessary.

The Droogenbroeck-Benedett Algorithm I encrypts at least 4-5 bitplanes out of eight bitplanes. Therefore, at least 50% of the bits (and likely more) undergo the encryption stage, and that is still a significant overhead. If the encrypted bitplanes are encrypted by means of a one-time pad, or simulate a one-time pad using a conventional stream cipher, the algorithm preserves a constant bitrate.

Table 5.1. Classification of Van Droogenbroeck-Benedett Algorithm I.

Domain	Uncompressed images
Perception-level	High
Cryptosystem	One-time pad or conventional stream cipher
Encryption Approach	Selective
Format Compliant	--
Bitrate	Constant

5.1.2 Van Droogenbroeck-Benedett Algorithm II

The second method by Van Droogenbroeck and Benedett is designed to selectively encrypt the JPEG compressed images. The Van Droogenbroeck-Benedett Algorithm II is based on encrypting certain discrete cosine transform (DCT) coefficients. Even though it was designed for JPEG images, the algorithm is most likely applicable to any DCT-based coded images.

In JPEG, the Huffman coder aggregates zero coefficients into runs of zeros, which results in the efficient compression. More precisely,

JPEG encoder creates symbols consisting of the run of zeros and the magnitude categories of the non-zero DCT coefficients that terminate the run of zeros. The Huffman stage of JPEG assigns codewords to these symbols. The codewords are then followed by the appended bits that fully specify the sign and magnitude of the non-zero coefficients that terminate the run of zeros. The Van Droogenbroeck-Benedett Algorithm II encrypts only these appended bits, while Huffman codewords are left unencrypted. However, not all non-zero DCT coefficients undergo such transformation. In [31], Van Droogenbroeck and Benedett suggested two different encryption modes:

- *Mode 1* - all coefficients are encrypted except for the DC coefficient,
- *Mode 2* - all coefficients are encrypted except for the DC coefficient and the first five AC coefficients.

Clearly, the first mode is more secure, but it involves a slightly longer processing time. On the other hand, the second mode leaves out quite a bit of visually important coefficients so that the level of visual degradation is minimal. The DC coefficients and the first few AC coefficients are left unencrypted, since their values are highly predictable anyway [31]. To measure the level of degradation it is a good idea to use standard image quality metrics such as the mean square error (MSE) and the peak signal-to-noise ratio (PSNR). Figure 5.3 shows the effects of two modes of Van Droogenbroeck-Benedett Algorithm II applied on the grayscale "Lena" image.

Figure 5.3. (a) The original uncompressed grayscale "Lena", (b) ordinary JPEG decoded "Lena", [MSE=4.07112, PSNR=42.0337], (c) JPEG decoded "Lena" encrypted with Van Droogenbroeck-Benedett Algorithm II (mode 1), [MSE=631.378, PSNR=20.1279], and (d) JPEG "Lena" encrypted with Van Droogenbroeck-Benedett Algorithm II (mode 2), [MSE=158.789, PSNR=26.1226].

As noted earlier, Van Droogenbroeck-Benedett Algorithm II achieves degradation of images rather than secure encryption of them. The result of this algorithm is an image of degraded visual appearance, which is exactly what is needed in particular applications. When a conventional cryptosystem is used, the cryptanalysis based on statistical analysis is hard to perform on less significant AC coefficients since these are less predictable. DC coefficients are highly predictable and they are left unencrypted. Since DC coefficients carry most information about the image, leaving these coefficients unencrypted causes image degradation rather than secure encryption. For high-security applications, this is of course not acceptable method.

Although, the actual percentage of coefficients selected for encryption varies from image to image, it is usually quite small. The Van Droogenbroeck-Benedett Algorithm II is fully format compliant, and it preserves the constant bitrate.

Table 5.2. Classification of Van Droogenbroeck-Benedett Algorithm II.

Domain	JPEG images or any DCT-based images
Perception-level	High
Cryptosystem	One-time pad or conventional stream cipher
Encryption Approach	Selective
Format Compliant	Yes
Bitrate	Constant

5.2 IMAGE ENCRYPTION BY PODESSER, SCHMIDT AND UHL

In [32], Podesser, Schmidt and Uhl proposed a selective encryption algorithm for the uncompressed (raster) images, that is quite opposite from the first method by Van Droogenbroeck and Benedett discussed above. The goal of this method is to provide high image security rather than image degradation.

5.2.1 Podesser-Schmidt-Uhl Algorithm

An 8-bit grayscale raster image contains exactly 8 bitplanes. As discussed in the previous section, the most significant bitplanes contains most visual information about the image, while less significant bitplanes contain information about fine details of the image (see Figure 5.1). After performing the experiments on raster images, Podesser, Schmidt and Uhl came to the same conclusion in [32] as Van Droogenbroeck and Benedett did in [31]. Namely, encrypting only the most significant bitplane is not secure since this bitplane is highly predictable. However, for applications where encryption must severely alienate the image as opposed to just visually degrading it, Van Droogenbroeck and Benedett Algorithm I is not suitable. On the other hand, Podesser-Schmidt-Uhl Algorithm can provide more severe image degradation.

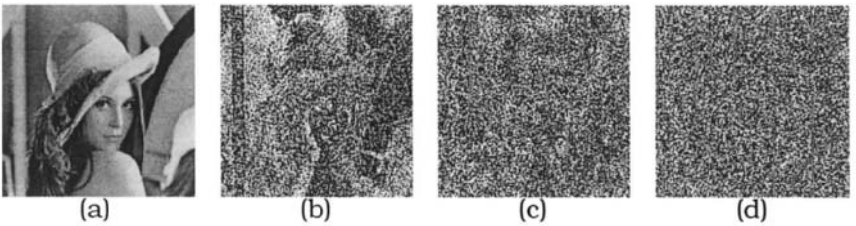
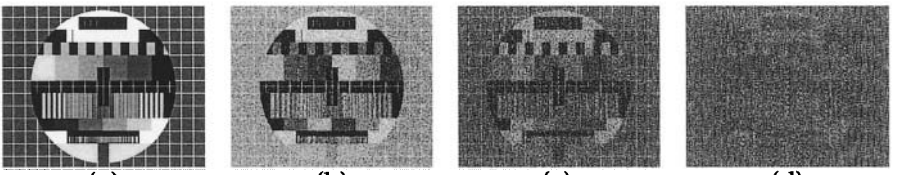


Figure 5.4. Podesser-Schmidt-Uhl Algorithm with natural images: (a) The original uncompressed grayscale "Lena", (b) "Lena" with encrypted MSB, (c) "Lena" with encrypted two MSB's, and (d) "Lena" with encrypted four MSB's.

The authors argued that most significant bitplane could be reconstructed with the aid of unencrypted remaining bitplanes, and therefore encrypting only the most significant bitplane is not sufficient. However, encrypting at least 2, and preferably 4 bitplanes leads to necessary security [32]. It is up to the user to determine whether the degradation method by encrypting only two bitplanes is sufficient for the given application, or more secure approach of encrypting four bitplanes is needed. The authors also noted that one should select the bitplanes of higher-order of significance to achieve more security, and that an additional security is achieved by not disclosing the information on which bitplanes were encrypted. Figure 5.4 demonstrates Podesser-Schmidt-Uhl Algorithm in action. Observe that if one chooses to encrypt four least significant

bitplanes, the effect is the same as the effect of the Van Droogenbroeck-Benedett Algorithm I.



(a) (b) (c) (d)

Figure 5.5. Podesser-Schmidt-Uhl Algorithm with high redundancy images: (a) The original uncompressed 1024×819 grayscale image "Test", (b) "Test" with encrypted MSB, (c) "Test" with encrypted MSB and the next bitplane, and (d) "Test" with encrypted MSB and three next bitplanes.

However, if the images contain pixel blocks with high redundancy, encrypting only two bitplanes preserves a lot of perceptual information (see Figure 5.5). Figure 5.5-c shows that in such images, even with four most significant bitplanes encrypted, one can still vaguely see the major circular shape that dominates the original image.

(a) (b) (c)

Figure 5.6. Replacement attack on Podesser-Schmidt-Uhl Algorithm: (a) the original image "Lena", (b) reconstructed "Lena" from a 25% encrypted image, and (c) reconstructed "Lena" from a 50% encrypted image (© 2002 IEEE).

The authors have analyzed the security of their algorithm and its vulnerability to the so-called *replacement attack* [32]. To perform a replacement attack, the adversary keeps the unencrypted bitplanes, and replaces the encrypted bitplanes with zero values. Since this

approach decreases the average luminance of the image, the pixel values are increased by a constant value, which depends on the number of encrypted bitplanes. If only MSB was encrypted, a value of 64 is added to each pixel. If MSB and the next bitplane were encrypted, a value of 96 is added (64 for the MSB and 32 more for the next bitplane) and so on.

The reconstructed image usually reveals a lot of visual information if only one, two, or three bitplanes are encrypted (see Figure 5.6-b). On the other hand, if four or more bitplanes were encrypted, reconstruction attack gives little or no information about the original image, as seen in Figure 5.6-c.

A second type of attack, called the *reconstruction attack* [32], tries to reconstruct the bitplanes of higher-order of significance, such as MSB, by using the information form unencrypted bitplanes. The attack exploits a well-known fact that, except for the edges, most regions of real-life images have large areas with smooth value changes. For such areas, pixels usually have the identical most significant bit values. One can divide an image into 2×2 pixel areas in which all 16 possible combinations of MSB configurations could be easily tested. Since we expect to have small differences between the pixels within a 2×2 block, we select the configuration with the smallest pixel differences. This approach could be used to try reconstructing the MSB. In many instances, this reveals a lot of visual information (see Figure 5.7).

(a) (b) (c)

Figure 5.7. Reconstruction attack on Podesser-Schmidt-Uhl Algorithm: (a) the original image "Lena", (b) the original most significant bitplane, and (c) reconstructed most significant bitplane (© 2002 IEEE).

However, the complexity of the reconstruction attack significantly increases when more than one bitplane is encrypted. In addition, the perception results are considerably reduced. The computational complexity of Podesser-Schmidt-Uhl algorithm is similar to that of the naïve approach when most bitplanes are encrypted. For high-security applications, where the visual perception must be minimal, at least four image bitplanes must be encrypted. Consequently, at least 50% of the data undergoes the encryption process, such as AES, in a stream cipher mode. Unfortunately, such performance is unacceptable in many applications.

Table 5.3. Classification of Podesser-Schmidt-Uhl Algorithm.

Domain	Uncompressed images
Perception-level	Variable
Cryptosystem	Conventional stream cipher
Encryption Approach	Selective
Format Compliant	Yes
Bitrate	Constant

5.3 IMAGE ENCRYPTION BY TANG

In 1996, Tang proposed one of the first selective encryption methods for DCT compressed images and videos [33]. The method is based on permuting the zigzag reordering, which is one of the stages in JPEG and MPEG algorithms. Unfortunately, Qiao and Nahrstedt showed in [34,35] that Tang overestimated the security level of his method. Nevertheless, we present the Tang Algorithm, which today represents an interesting case study.

5.3.1 Tang Algorithm

The JPEG images and the I-frames of MPEG video undergo a zigzag reordering of the 8×8 blocks. The zigzag pattern forms a sequence of 64 entries that are ready to enter entropy-encoding stage. The main idea of this approach is to use a random permutation list to map the individual 8×8 blocks to a 1×64 vector.

Tang performed a series of experiments that showed the following properties of JPEG encoded images. First, if one permutes all of the

AC coefficients within the image, the content of the image is still visible. This means that if the attacker places the DC coefficient into its actual position, which is the first position in the block, the image becomes recognizable, which is unacceptable for high security applications. Therefore, the knowledge of the position of the DC coefficient is important. Another important observation is that if we set the last AC coefficient to a random value for each block, the degradation of image quality is negligible. These observations point out that blindly applying the random permutation leads to a weak result. Since DC coefficients usually have the highest value among other DCT coefficients within the block, they are highly recognizable. Thus, it is a good idea to somehow reduce the value of DC coefficients before applying a random permutation to the sequence. Tang Algorithm, which is presented next, does exactly that.

Tang Encryption Algorithm

INPUT: Uncompressed raster image I.
OUTPUT: Encrypted image I' in JPEG format.

1. Generate a random permutation π of degree 64
2. For each 8×8 block B in I do the following
    a. Proceed with JPEG transformations of B until the quantization stage is finished.
    b. Set $b_7b_6b_5b_4b_3b_2b_1b_0$ to the 8-bit representation of DC coefficient of B
    c. Set the DC coefficient of B to $b_7b_6b_5b_4$
    d. Set the AC_{63} (the last AC coefficient) to $b_3b_2b_1b_0$
    e. Permute the coefficients in a 1-D representation of B according to the permutation π
    f. Continue with the JPEG algorithm for B
    g. Place the final symbols into the JPEG stream I'
3. Output I'

The decryption is a straightforward application of the inverse transformations, assuming that the same secret permutation (or its inverse permutation) is known. Tang Algorithm for image encryption tries to disguise the DC coefficient for each block by splitting its bits into equal halves, and then placing them into the actual DC coefficient place and the actual AC_{63} coefficient place. Then the list is permuted according to a secretly generated permutation. This

transformation is repeated for each block, and the secret permutation is kept fixed. Figure 5.8 shows an encryption of "Lena" using Tang Algorithm. Comparing the values of MSE and PSNR of Figure 5.8-b and Figure 5.8-d shows how minimal the loss of quality is when the last AC coefficient is randomized, which happens due to the encryption algorithm. In addition, the compression ratio (CR) is visibly smaller due to the inefficient reordering.

(a) (b) (c) (d)

Figure 5.8. Tang Algorithm with natural images: (a) The original uncompressed grayscale "Lena", (b) regular JPEG "Lena" [MSE=29.6917, PSNR=33.4045, CR=24.2129], (c) encrypted "Lena" with Tang Algorithm as decoded by an ordinary decoder [MSE=33404.8, PSNR=2.89271, CR=6.83922], and (d) properly decrypted "Lena" [MSE=30.0801, PSNR=33.348, CR=6.83922].

Unfortunately, the Tang algorithm is not particularly secure [34,35]. Qiao and Nahrstedt introduced two types of attacks on zigzag permutation: a chosen/known-plaintext attack, and a ciphertext-only attack. A chosen/known-plaintext attack is particularly applicable to the videos with known clips such as blank screens, the movie rating screens, MGM roaring lion, etc. If these known frames are used as a comparison, the adversary can easily recover the secret permutation π, and completely break the system. Since Tang himself realized the obvious vulnerability to a chosen/known-plaintext attack, he introduced an improvement of the zigzag permutation technique by using the binary coin flipping sequence method together with two different permutations of degree 64. Two permutations π and σ are generated and for each 8×8 block a coin is randomly flipped. The outcome event determines which permutation to apply. If the head is flipped, we apply π, and if the tail is flipped, we apply σ. As shown in [34,35], this addition turns out to be useless. If we know some of the original frames in advance (known-plaintext) then by simple comparison both permutation lists can be found. Because of the certain statistical properties of the DCT

blocks (upper left corner gathering of the AC coefficients within a block), we can easily determine which permutation is used for each block. In addition, Qiao and Nahrstedt showed that the Tang's splitting method is just a subset of previously proposed MPEG encryption method by Meyer and Gadegast called SECMPEG, which was known to have security issues (to be discussed in the next chapter). Finally, Qiao and Nahrstedt showed that the Tang's zigzag permutation algorithm is susceptible to the ciphertext-only attack. The attack relies on the statistical properties of DCT coefficients where most non-zero terms are gathered in the top left corner of the block. Statistical analysis shows the following observation [35]. The DC coefficient always has the highest frequency value, the frequencies of AC_1 and AC_2 coefficients are among top 6, and frequencies of AC_3 and AC_4 are among top 10. Therefore, one can reconstruct the original video with a relatively good accuracy using only five DCT coefficients. In that respect, Tang's zigzag permutation cipher seriously lacks the desired level of security.

Finally, since the zigzag reordering is permuted, there is an expected loss in the compression ratio. The reason why we reorder the coefficient in a zigzag fashion is to create a long zero runs, and permuting a sequence would likely decrease the zero runs needed for efficient Huffman compression. In the example from Figure 5.8, the compression ratio ended up being more than 3.5 times worse with Tang's encryption than without it. Finally, the Tang Algorithm itself is not fully defined cryptosystem, since the key setup stage was never actually specified by the author. In the standard cryptographic setting, the random permutation(s) must be somehow generated for some given secret *n*-bit key. The computational complexity of this stage should be considered when judging the overall performance of the algorithm.

Table 5.4. Classification of Tang Algorithm.

Domain	JPEG images or any DCT-based images
Perception-level	Medium
Cryptosystem	Pseudo-random permutation
Encryption Approach	Selective
Format Compliant	Yes
Bitrate	Smaller (bigger size)

5.4 IMAGE ENCRYPTION BY SHI, WANG AND BHARGAVA

In [38], Shi Wang and Bhargava classified their previous work from [36,37] into four different MPEG video encryption algorithms. However, only the first algorithm from [38] is suitable for both images (JPEG) and videos (MPEG I-frames).

5.4.1 Shi-Wang-Bhargava Algorithm I

The first algorithm by Shi, Wang, and Bhargava, denoted here by Shi-Wang-Bhargava Algorithm I, uses the permutation of Huffman codewords in the JPEG images. The secret part of the algorithm is a permutation π, which is used to permute the standard Huffman codeword list. In order to save the compression ratio, π must be such that it only permutes the codewords with the same number of bits. In addition, the distance between the original and the permuted codeword list must be greater than the encryption quality parameter b, which is chosen by the user.

The security of Shi-Wang-Bhargava Algorithm I is not particularly good. In [39], it is shown that Shi-Wang-Bhargava Algorithm I is highly vulnerable to both known-plaintext attack, and ciphertext-only attack. If some of the video frames are known in advance (such as standard introductory jingles and similar), one can reconstruct the secret permutation π by comparing the original and encrypted frames. Vulnerability to this type of attack was also discussed in [38]. However, the algorithm is also subject to ciphertext-only attack. The low-frequency error attack can be applied on ciphertext produced by Shi-Wang-Bhargava Algorithm I [39]. Since the permutation π is of the special form, it only shuffles codewords with the same length. Therefore, the most security comes from shuffling the 16-bit codewords in the AC coefficient entropy table. However, since there is a very limited number of codewords with length of less than 16 bits, it is very easy to reconstruct all of the DC coefficients and most frequent AC coefficients (since these will be encoded with less than 16-bit codewords). In other words, the only hard part would be to figure out how the permutation π shuffles the 16-bit codewords. However, these are appearing extremely rare, and the reconstructed image may be of almost the same quality as the original. Furthermore, the rare pixels that do contain 16-bit

codewords can easily be interpolated to get rid of the noise effect and to create the reconstruction image that, at least to the human visual system, appears identical to the original.

Another drawback is that Shi-Wang-Bhargava Algorithm I is not format compliant. The decoders would crash since the codeword list is out of order. Finally, the way the algorithm generates the random permutation was not fully specified.

Table 5.5. Classification of Shi-Wang-Bhargava Algorithm I.

Domain	JPEG images or any DCT-based images, I-frames of MPEG videos
Perception-level	--
Cryptosystem	Restricted pseudo-random permutation
Encryption Approach	Selective
Format Compliant	No
Bitrate	Constant or near-constant

5.5 IMAGE ENCRYPTION BY CHENG AND LI

In 2000, Cheng and Li [40] proposed selective encryption methods that are suitable for images compressed with two specific classes of compression algorithms: (1) quadtree compression algorithms, and (2) wavelet compression algorithms based on zerotrees.

5.5.1 Cheng-Li Algorithm I

Before introducing the first image encryption algorithm proposed by Cheng and Li, we need to cover some basics of quadtree image compression. In general, a quadtree can be utilized to produce an elegant image compression technique. The quadtree compression performs well for the low bitrates, and it does not require an extensive computational power. As such, it is suitable for many applications related to small portable devices.

Quadtree Compression of Images

A *quadtree* is a rooted tree where every node has either zero or four children. A node with four children is called an *internal node*, while a node with zero children is called a *leaf node*. A *level* of a node is the number of edges in the shortest path from the root to that node, and the largest level within the quadtree is called the *height*. Figure 5.9 shows a simple quadtree with its parameters that we just defined.

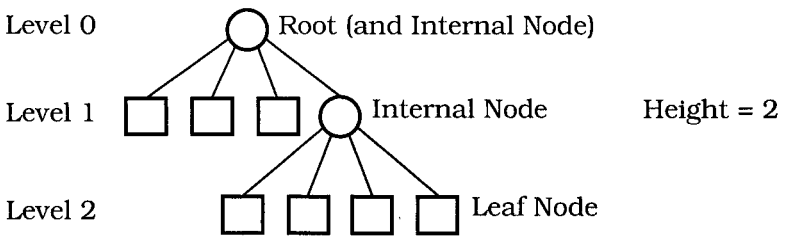


Figure 5.9. A quadtree of height 2 with 2 internal nodes and 7 leaf nodes.

Quadtree structure can be used for either lossless or lossy compression of images. Essentially, the algorithm recursively creates a quadtree structure according to the color values or similar parameters. All leaf nodes of a quadtree carry a corresponding parameter value. Without loss of generality, the only parameter we consider is the color value. At the end of the algorithm, two binary sequences are created: the tree structure sequence, and the sequence containing the color values. If the entire image has the same color, the quadtree contains only the root node with the corresponding color value. Otherwise, the root node becomes an internal node, and the image is split into four quadrants.

Each quadrant is then examined the same way and recursively, until the entire image is examined and the two sequences created. This is how lossless quadtree image compression works. However, there are many ways one can reorder the nodes within the tree, and also many ways one can reorder the nodes when they are represented in a sequence. We consider a standard way to reorder the child nodes within the quadtree, which is NW, SW, SE, and NE in that order, but different implementations may use different orderings. As far as the

leaf ordering in the sequence of color values, the following two orderings are usually used: Leaf Ordering I, and Leaf Ordering II.

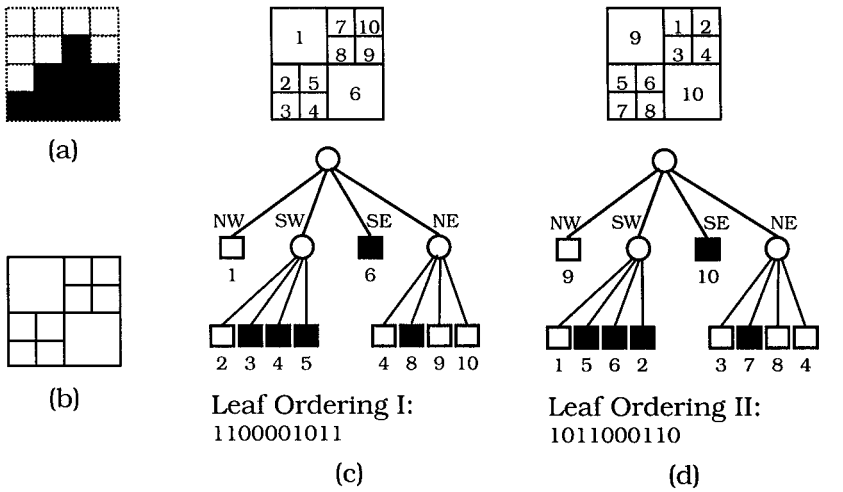
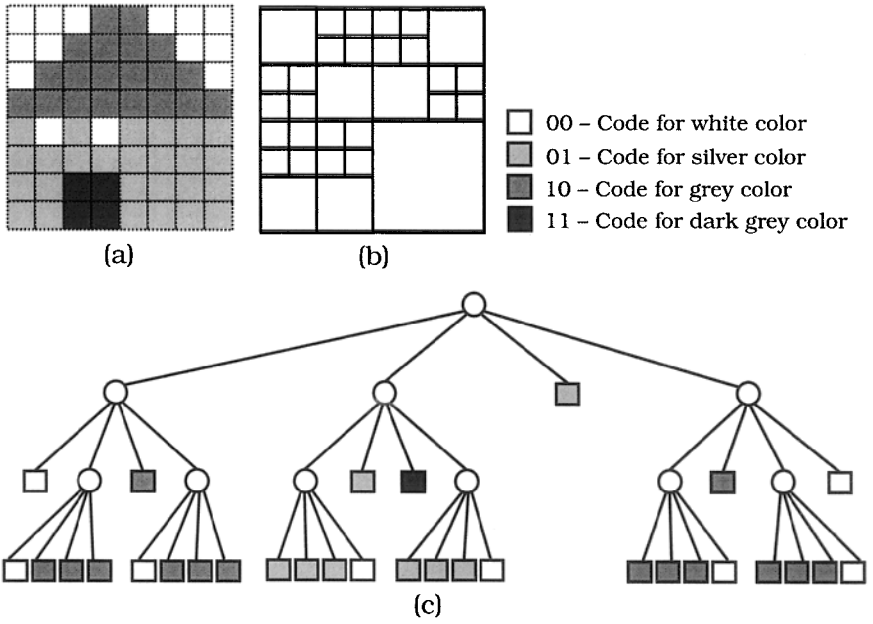


Figure 5.10. Representing a binary image using a quadtree: (a) the original image, (b) the quadtree decomposition, (c) ordering leaf nodes in the quadtree using Leaf Ordering I, and (d) ordering leaf nodes in the quadtree using Leaf Ordering II and *bottom-up* implementation.

Leaf Ordering I is made by the inorder traversal of the quadtree [40]. For a given quadtree where child nodes are corresponding to the NW, SW, SE, and NE quadrants in that order, Leaf Ordering I takes leaf nodes left to right, regardless of the tree level. Figure 5.10-c shows how the leaf nodes are ordered using Leaf Ordering I. In this example, 0 represents a black pixel, while 1 represents a white pixel. In addition, a circular shape represents an internal node, while a rectangular shape whose color matches to the color value of the corresponding quadrant represents a leaf node.

On the other hand, Leaf Ordering II is heavily based on the tree levels. In this ordering, the leaf values are encoded one level at a time. One type of implementation starts from the lowest tree level, and ends with the highest. This is referred to as the *top-down* implementation. An opposite implementation is the so-called *bottom-up* implementation, which starts from the smaller levels and moves to the highest. In any case, the leaf nodes within the same

level are ordered by the regular raster scan, left-to-right and top-to-bottom (see Figure 5.10-d). In many instances, the bottom-up implementation is more suitable for the lossy quadtree compression, where it allows better control over the compression ratio, and even a better performance when used together with Leaf Ordering II.



Sequence #1 (the tree structure):
0001010100110010111111111111111111111111

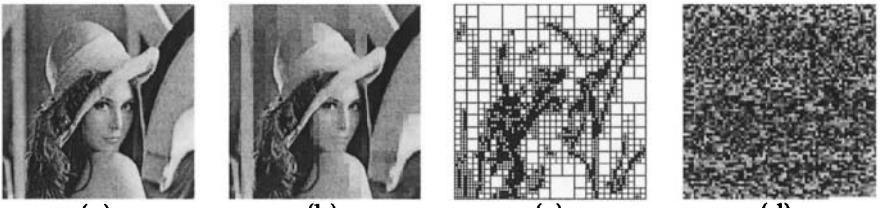
Sequence #2 (the color values):
0100001010011100101000101010100010100010101010010001000101010 1

Figure 5.11. Compressing an 8×8 4-color image using a lossless quadtree compression: (a) the original image, (b) quadtree decomposition, and (c) the resulting quadtree with color values and the two final sequences.

An example of a lossless quadtree compression of simple 8×8 4-color image is illustrated in Figure 5.11. The 8×8 quaternary image from Figure 5.11 takes a total of 128 bits when stored in an ordinary bitmap representation, since there are 64 pixels and each pixel color value uses two bits.

Using quadtree compression, the compressed image from above takes only 103 bits (41 bits to store the quadtree structure, and 62 bits to store the color values). The reconstruction of the image for the given sequences is straightforward. The first sequence uniquely defines the quadtree structure, as shown in Figure 5.11-b. Whenever zero appears in the sequence, the node is an internal node, and whenever one appears, the node is a leaf node. It is assumed that the node ordering corresponds to the order NW, SW, SE, and NE. Once the quadtree structure is determined, the next is to fill in the color information. This is easily accomplished by using the second sequence, where each binary codeword corresponds to a particular color. In Figure 5.11, each codeword was exactly two bits long, and each codeword specified exactly one out of four different colors. We start filling the largest squares, moving towards smaller squares, and ending with the smallest squares, which possibly have a size of a pixel. In other words, filling starts from the lowest tree level, and ends with the highest (*top-down* implementation).

Obviously, knowing the leaf ordering and the implementation type is necessary in order to decode the compressed image. Figure 5.12 illustrates the quadtree compression of image "Lena".



(a) (b) (c) (d)

Figure 5.12. Compressing image "Lena" using quadtree compression: (a) the original image, (b) the compressed image, (c) the quadtree decomposition, and (d) the sequence of color values for the *bottom-up* implementation with Leaf Ordering II.

The security of the selective encryption method for quadtree images proposed by Cheng and Li relies on the fact that one cannot reconstruct the original image (or larger pieces of it) by simply observing the color values sequence and not having the exact quadtree decomposition. As long as the sequence of color values is not correlated to the original image or to the quadtree decomposition, the algorithm provides strong security. However, as noted in [40], one must be very careful when choosing the leaf

ordering, since different ordering results in different degree of correlation.

In general, trying to exhaustively search for the unknown quadtree decomposition is out of the question for most images. For the lossless compression, the number of levels can be computed from the number of leaf values, which is bad since the number of leaf values is always known to the attacker [40]. For the lossy compression, that value can be approximated by good bounds [40]. However, the amount of possible quadtrees that satisfy the given number of leaf nodes and the given tree height is very large, except for the rare cases when an image generates almost empty or almost complete quadtree.

However, selective encryption of quadtree compression that utilizes Leaf Ordering I has some undesired properties in respect to the security. One of the properties is that the leaf values of the sibling nodes are close to one another in the encoded sequence. Notice that the number of bits used for encoding each leaf value (color) is fixed, so that it is easy to separate individual leaf values from the unencrypted binary sequence. Clearly, if the NW, SW, SE, and NE quadrant ordering was used, then the first leaf value corresponds to the NW corner of the image and the last leaf value corresponds to the NE corner of the image. Another undesired property is that subsequences of identical values can be used for further cryptanalysis. Having four identical values in a row implies that these nodes are not siblings, since otherwise they would have been merged into a single node. These properties could be exploited to gain information about the image size, compression ratio, and the image histogram [40]. Although it is highly unlikely that these attacks will reveal the entire compressed image, they may reveal homogeneous regions at the corners and may reveal the shapes of some foreground objects.

On the other hand, Leaf Ordering II is more resilient to cryptanalysis. In this type of ordering sibling nodes could be far apart, and thus, it is not possible to make use of the fact that four sibling nodes cannot have the same value. In addition, the first and the last leaf values do not necessarily correspond to the corners of an image. Therefore, the authors suggest using Leaf Ordering II rather than Leaf Ordering I to ensure higher security level.

The algorithm selects only the binary sequence corresponding to the quadtree decomposition. Although the quadtree decomposition depends on the actual image and whether a lossless or lossy compression is applied, the size of this sequence is relatively small in comparison to the sequence containing the leaf values. The experiments show that quadtree decomposition usually constitutes roughly 13%-27% of the bitstream [40]. Therefore, only about a quarter of the entire compressed bitstream needs to be encrypted using a conventional cryptosystem, resulting in approximately four times faster performance of the encryption stage than that of the naive approach. Clearly, the algorithm is not format compliant.

Table 5.6. Classification of Cheng-Li Algorithm I.

Domain	Quadtree compressed images
Perception-level	--
Cryptosystem	Conventional
Encryption Approach	Selective
Format Compliant	No
Bitrate	Near-constant

5.5.2 Cheng-Li Algorithm II

The second encryption algorithm proposed by Cheng and Li [40] is targeted towards the images compressed with zerotree wavelet compression. To understand the Cheng-Li Algorithm II, we need to present a preliminary overview of zerotree wavelet image compression.

Zerotree Wavelet Compression

The discrete wavelet transform (DWT) has been recently very successfully applied for image and video compression. This kind of transform provides the time-frequency representation of the raw signal. Essentially, DWT decomposes a non-stationary signal (such as the real-life image or video) into a set of multilevel subbands. In fact, DWT is often referred to as the *hierarchical subband decomposition*, or *pyramid decomposition*. Each component of a subband becomes more stationary and hence becomes easier to process. Wavelet-based coding provides significant improvements in picture quality at low bitrates. In addition, it is computationally

efficient and it offers enhanced features such as scalability. Even the JPEG2000 and MPEG-4 standards make use of the wavelet-based compression algorithms for coding still texture and images. Recently, a variety of powerful and sophisticated wavelet-based image compression algorithms have been developed. The two important coding schemes that have emerged are: (1) embedded zerotree wavelet (EZW) coding by Shapiro [41], and (2) set partitioning in hierarchical trees (SPIHT) coding by Said and Pearlman [42]. Both EZW method and SPIHT algorithm belong to the category of embedded algorithms. In an *embedded algorithm*, the encoder can terminate the encoding process at any stage to allow the target bitrate. In addition, the decoder can stop decoding at any stage, and the resulting image would correspond to the same image that would be encoded at the corresponding bitrate.

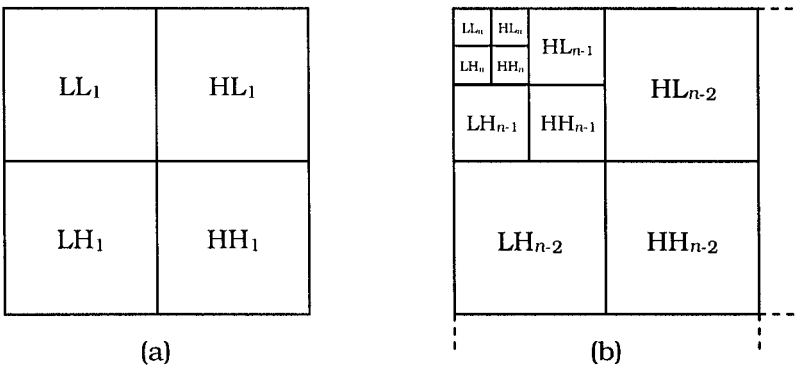


Figure 5.13. Subbands created by discrete wavelet transforms at (a) the first stage, and (b) the n^{th} stage.

Cheng and Li focused on SPIHT algorithm, but in general, this is just an improvement of the original EZW coding, and it is also a base for many other wavelet-based image compression algorithms. In the first stage of a discrete wavelet transform the image is divided into four subbands denoted by LL_1, LH_1, HL_1, and HH_1, as shown in Figure 5.13-a. The four subbands are result of two filters, vertical and horizontal, which are applied to an image. Issues related to designing and choosing such filters are beyond the scope, and an interested reader should refer to [43]. The subbands LH_1, HL_1, and HH_1 correspond to the finest level of wavelet coefficients. The subband LL_1 is further divided into four subbands LL_2, LH_2, HL_2, and HH_2. This process recursively continues until some final level n

(see Figure 5.13-b). Subbands at n^{th} level represent the coarsest level of wavelet coefficients, and it is often referred to as the *root level.*

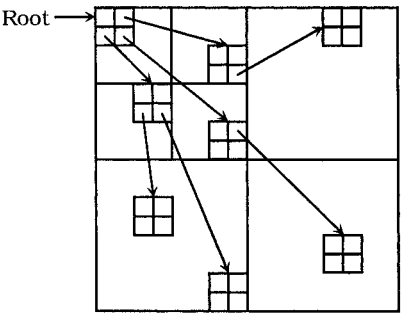


Figure 5.14. Spatial orientation tree structure defined over DWT coefficients of a sample image. The arrows are directed from a node n to a 2×2 submatrix whose four nodes are an offspring of n.

The zerotree-based coding concept was first introduced by Shapiro [41]. In general, there is often a correlation between coefficients in different levels of subbands. Zerotree coding provides an efficient multi-resolution representation of binary maps called *significance maps*, which indicate the positions of significant coefficients. This allows successful prediction of insignificant coefficients across different levels. *Zerotree* is simply a rooted tree whose nodes represent certain DWT coefficients. For a given threshold T a coefficient x is *insignificant* if $|x| < T$. Here, $|x|$ denotes the magnitude of x. Otherwise, x is *significant.* A coefficient x is an element (node) of a zerotree if itself and all of its descendants are insignificant with respect to T. Using only few encoding symbols, the zerotree coding can efficiently encode the significance map by predicting the absence of significant information across levels [41]. In [42], Said and Pearlman developed the SPIHT algorithm (an acronym for "set partitioning in hierarchical trees"), which further improves the performance of Shapiro's EZW method. The SPIHT algorithm is similar to Shapiro's approach, however, a different logical structure is defined that can be efficiently utilized.

A usual discrete wavelet transforms are performed and an image is recursively split into subbands, thus forming a hierarchical pyramid. Said and Pearlman defined *spatial orientation tree*, which naturally

identifies the spatial relationship on this hierarchical decomposition. The nodes of this tree simply correspond to the DWT coefficients from the hierarchical decomposition. Since the coefficients are logically grouped in a matrix-like form, a node is uniquely identified by the coordinates of corresponding coefficient. A coefficient at (i, j) is denoted by $c_{i,j}$. In addition, the spatial orientation tree has a quadtree structure; i.e., each node has either zero or four children (offspring). However, the four offspring nodes of a given parent (descendant node) at a level i are such that they form a 2×2 submatrix of coefficients within a subband located at level $i-1$ (see Figure 5.14).

In such a setting, the following sets of coordinates are defined:

- $O(i, j)$ – coordinates of all offspring of the node (i, j)
- $D(i, j)$ – coordinates of all descendants of the node (i, j)
- $L(i, j) = D(i, j) - O(i, j)$
- H – coordinates of all root nodes

It is easy to observe that in case of a spatial orientation tree, either $O(i, j) = \varnothing$ or $O(i, j) = \{(2i, 2j), (2i, 2j + 1), (2i + 1, 2j), (2i + 1, 2j + 1)\}$.

Both EZW and SPIHT methods are based on the following three concepts: 1) partial ordering of wavelet coefficients by their magnitude, 2) ordered bitplane transmission of refinement bits, and 3) exploitation of the correlation between wavelet coefficients across different scales. However, in SPIHT algorithm, the way the coefficients are partitioned in the sorting (ordering) algorithm is fundamentally different from that of the EZW method, and it depends on the aforementioned spatial orientation trees.

The beauty and efficiency of SPIHT coding algorithm is that the ordering information does not need to be transmitted. Instead, the ordering information can be recovered from the execution path. Namely, an execution path of an algorithm can be determined by simply knowing the comparison results at its branching points. Therefore, if the decoder uses the same sorting algorithm, the original ordering can be recovered if the encoder's comparison results are known. This feature improves the compression ratio since the transmission of comparison results is much cheaper than the transmission of entire ordering information. In addition, not all coefficients need to be sorted (ordered). The sorting algorithm

divides the nodes into the partitioning subsets. For each subset τ the following Boolean value $S_n(\tau)$ is calculated:

$$S_n(\tau) = (\max_{(i,j)\in\tau}\{|c_{i,j}|\} \geq 2^n).$$

If $S_n(\tau)$ is 0 (False), then all coefficients corresponding to τ are insignificant. However, if $S_n(\tau)$ is 1 (True), then the set τ is further partitioned into subsets that undergo the same test.

The SPIHT encoding algorithm consists of four conceptual stages: 1) initialization, 2) sorting pass, 3) refinement pass, and 4) quantization-step update. For efficiency, the encoding algorithm maintains three ordered control lists: the list of insignificant sets (LIS), the list of insignificant pixels (LIP), and the list of significant pixels (LSP). The elements of LIP and LSP are pixels represented by their coordinates (i, j), while the elements of LIS are sets, either $D(i, j)$ or $L(i, j)$. A LIS entry is of type A if it represents $D(i, j)$, and of type B if it represents $L(i, j)$. Initially, the LSP is an empty list, the LIP contains the coordinates from H, while the LIS contains the coordinates from H with descendants, as type A entries. During the sorting pass, the pixels in the LIP (which were insignificant in the previous pass) are tested and the significant ones are moved to the LSP. Next, the sets from LIS are sequentially tested, and significant ones are removed and further partitioned. If a new subset obtained by this partition contains more than one element, then the subset is added back to the LIS. On the other hand, single-coordinate sets are added to the end of LIP or LSP, depending on whether they are significant or insignificant. The coordinates from LSP are used in the refinement pass stage. For completeness, we present the pseudo code of SPIHT algorithm.

SPIHT Encoding Algorithm

INPUT: Wavelet coefficients of a raw image
OUTPUT: SPIHT Encoded bitstream

1. (*Initialization*): Compute $n = \lfloor \log_2(\max_{(i,j)}\{|c_{i,j}|\})\rfloor$. Set the LSP to an empty list. Add the coordinates from H to the LIP, and only those with descendants to the LIS, as type A entries. Write n to the output.
2. (*Sorting Pass*):

For each $(i, j) \in$ LIP do the following:

a) Write $S_n(\{(i, j)\})$ to the output.

b) If $S_n(\{(i, j)\}) = 1$ then move (i, j) to the LSP and write the sign of $c_{i,j}$ to the output.

For each $(i, j) \in$ LIS do the following:

c) If (i, j) is of type A then:

- Write $S_n(D(i, j))$ to the output.
- If $S_n(D(i, j)) = 1$ then:
  - For each $(k, l) \in O(i, j)$ do the following:
    - Write $S_n(\{(k, l)\})$ to the output.
    - If $S_n(\{(k, l)\}) = 1$ then add (k, l) to the LSP and write the sign of $c_{k,l}$ to the output.
    - If $S_n(\{(k, l)\}) = 0$ then add (k, l) to the end of LIP.
  - If $L(i, j) \neq \varnothing$ then move (i, j) to the end of the LIS, as an entry of type B; otherwise, remove entry (i, j) from the LIS.

d) If (i, j) is of type B then:

- Write $S_n(L(i, j))$ to the output.
- If $S_n(L(i, j)) = 1$ then:
  - Add each $(k, l) \in O(i, j)$ to the end of the LIS, as an entry of type A.
  - Remove (i, j) from the LIS.

3. (*Refinement Pass*): For each $(i, j) \in$ LSP, except those included in the last sorting pass (i.e., the ones with the same n), write the n^{th} most significant bit of $|c_{i,j}|$ to the output.
4. (Quantization-Step Update): If the desired bit rate is reached then stop; otherwise, decrement n by 1 and go to the step 2 (sorting pass).

Wavelet compression algorithm based on zerotrees, such as SPIHT, transmits the structure of the zerotree along with the significant coefficients. The SPIHT algorithm transmits the significance of the coefficient sets that correspond to the trees of coefficients. It uses the significance of the coefficient sets to reconstruct the zerotree structure. As we saw earlier, the structure of the tree strongly affects the execution of the algorithm. The SPIHT encoded image contains different types of data such as the sign bits, the refinement bits, and the significance bits (significance of pixels and significance of sets). Even the small amount of incorrect information at the

beginning of the encoded block containing the significance bits causes the failure in the decoding process. Incorrect significance bits often cause the misinterpretation of future bits, unlike the incorrect sign bits or refinement bits, which do not. With this in mind, Cheng and Li proposed encrypting only the initial significance information of pixels and sets, as well as the initial threshold parameter n that determines which coefficients are significant [40]. They restricted to encrypting only those significance bits, which are in the two highest pyramid levels, since all other pixels and sets are derived by decomposing the sets from the highest two pyramid levels. More precisely, the significance information $S_n(\{(i, j)\})$, $S_n(D(i, j))$, and $S_n(L(i, j))$ is encrypted only if (i, j) is in the two highest pyramid levels.

The information about the significant bits from the first two pyramid levels cannot be deduced just by observing the lower levels. On the other hand, this information is crucial to the decoder, and without it, the initial changes to the three lists LIS, LIP, and LSP are not known, and the wavelet coefficients cannot be reconstructed.

There are several reasons that make the Cheng-Li Algorithm II difficult for cryptanalysis. There are mixed types of bits in the encoded bitstream, however, without the significance information, it is hard to correctly determine their type at the decoder's side. In addition, the decoder would fail to locate the starting points of sorting and refinement passes. If the initial state of the LSP is not known, it is hard to determine the order in which the coefficients are examined in the refinement pass. Next, there are many-to-many relationships between important part and the unimportant part. Finally, the important parts of two similar images may be completely different. Although nobody should ever guarantee that a system is highly secure, there are no known successful attacks on the Cheng-Li Algorithm II, which makes it suitable for the secure image applications.

According to the experiments, the size of the important part (i.e., the part that is selected for encryption) is up to 2% of the bitstream for the 512×512 images, and up to 7% for the 256×256 images [40]. Therefore, the overall performance of Cheng-Li Algorithm II is excellent considering such a small overhead that still provides high security. In addition, the compression ratio is not affected by the algorithm; however, the algorithm is not format compliant.

Table 5.7. Classification of Cheng-Li Algorithm I.

Domain	Zerotree wavelet compressed images
Perception-level	--
Cryptosystem	Conventional
Encryption Approach	Selective
Format Compliant	No
Bitrate	Near-constant

5.6 IMAGE ENCRYPTION BY YEN AND GUO

Yen and Guo proposed a series of image encryption algorithm based on a one-dimensional chaotic map [44]. Chaotic-based encryption algorithms generated considerable attention due to their fast performance and suitability for digital multimedia. In fact, chaos-based algorithms have shown excellent properties in terms of security, complexity, speed, computing power and computational overhead. The algorithm proposed by Yen and Guo in [44], however, represents an oversimplified use of the chaotic approach, and consequently, it lacks security. The algorithm is very fast and not selective since the entire image is encrypted.

5.6.1 Yen-Guo Algorithm (CKBA)

In 2000, Yen and Guo proposed a fast algorithm for encrypting the uncompressed images. The Yen-Guo Algorithm I, also known as the chaotic key-based algorithm (CKBA), relies on a one-dimensional chaotic map for generating a pseudo-random key sequence. The algorithm belongs to the category of pixel value substitution ciphers, where the image pixel values are transformed by a pseudo-random key sequence.

The encryption of an $M \times N$ image I by CKBA is realized as follows. Without loss of generality, assume 8 divides $M \times N$. Select two secret 8-bit keys k_1 and k_2, and a secret 16-bit initial condition $x(0)$ of a

one-dimensional chaotic system. Iteratively run the chaotic system $MN/8 - 1$ times to produce a sequence of 16-bit numbers $\{x(i)\}$, $0 \le i < MN/8$, with $\{b(i)\}$, $0 \le i < MN/8$, being its binary representation. If $I(x,y)$ is an 8-bit pixel value in the plaintext image I, with $0 \le x < M$ and $0 \le y < N$, the corresponding ciphertext pixel $I'(x,y)$ is defined by the following rule:

$$I'(x,y) = \begin{cases} I(x,y) \oplus k_1, & \text{if } b'(x,y) = 3; \\ I(x,y) \oplus \overline{k}_1, & \text{if } b'(x,y) = 2; \\ I(x,y) \oplus k_2, & \text{if } b'(x,y) = 1; \\ I(x,y) \oplus \overline{k}_2, & \text{if } b'(x,y) = 0, \end{cases}$$

where $b'(x,y) = 2b(l) + b(l+1)$ and $l = 2(xN + y)$.

Even though the keys k_1 and k_2 are chosen at random, the authors have restricted that the Hamming distance between them be 4. In other words, k_1 and k_2 should differ at exactly four positions. A quick observation of the encryption scheme shows that the decryption process is the identical mapping since XOR is an involution. Figure 5.15-a and Figure 5.15-b shows an example of Yen-Guo Algorithm for encryption keys k_1=92, k_2=36, and $x(0)$=0.5876. In this example, as Yen and Guo did in the original experiments, we have used a one-dimensional chaotic map known as the *Logistic map.* The Logistic map is defines as follows:

$$x_n = \mu x_{n-1}(1 - x_{n-1}),$$

where $\mu \in (0,4]$, and $x \in (0,1)$.

For the example, we have chosen the parameter μ to be 3.95. It is worth mentioning here that the Logistic map behaves highly chaotically when the parameter μ is chosen to be greater than the accumulation point 3.569945672. Thus, all implementations of CKBA should choose μ from the interval (3.569945672, 4]. In addition, one could use other one-dimensional maps with CKBA. It is known that there are one-dimensional maps that behave more chaotically than the Logistic map, since the Logistic map lacks some important randomness properties, such as the balance property. For example, a better randomness behavior is achieved by the ICMIC map [45], and by the PWLCM map [46].

One problem with CKBA is the lack of security. The key search space of CKBA is only 2^{32} bit secret keys, which is very tiny by today's standards. Even worse, Li and Zheng showed in [47] that the actual search space of CKBA is only about 2^{30} due to the key restrictions.

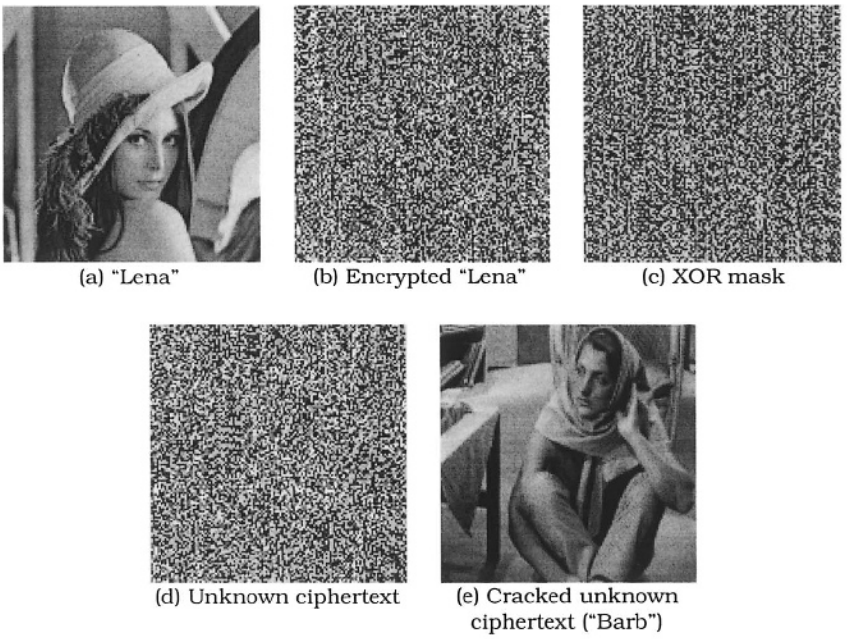


Figure 5.15. Chosen/known-ciphertext attack on CKBA: the attacker calculates (c) by XOR-ing (a) and (b), and then (e) by XOR-ing (c) and (d).

Furthermore, the original scheme is subject to well-defined chosen/known-plaintext attacks [47]. That is, CKBA can be completely broken if only one plaintext image and its corresponding ciphertext image are known. Suppose we have the images I and its CKBA encryption I' obtained by using secret keys k_1, k_2, and $x(0)$. By the definition of CKBA, I' can be obtained from I by XOR-ing it with a particular image mask I_m. Consequently, the image mask I_m can be obtained simply by XOR-ing images I and I'. This mask can then be used to completely decrypt all other images of same or smaller size for which the same keys k_1, k_2, and $x(0)$ were used. Figure 5.15 demonstrates chosen/known-plaintext attack on CKBA where the same key is used to encrypt both 128×128 image "Lena"

and 128×128 image "Barb". The attacker can easily recover "Barb" from the unknown ciphertext in Figure 5.15-d.

In addition, Li and Zheng constructed a $O(MN)$ brute force algorithm for the CKBA scheme that could be used to completely recover the keys k_1, k_2, and $x(0)$, provided that I_m is known, which makes it possible to recover all images encrypted with the same key, regardless of the image size [47].

An obvious improvement, also discussed in [47], is to increase the key space. Modifying CKBA so that it utilizes longer keys could improve its sustainability to a ciphertext-only attack. Yen and Guo have slightly improved the original CKBA scheme described above in further publications, [48,49], however, all of these schemes remain vulnerable to the chosen/known-plaintext attacks [7,50].

In conclusion, the Yen-Guo Algorithm and their successors are very fast, and suitable for hardware easy and cheap implementations. The downside is an obvious lack of security. Note that the security of CKBA was vastly overestimated by its authors in [44], and that CKBA was intended for high-security applications, such as medical imaging or military image databases. Due to lack of security, CKBA should not be used in such high security environments.

Table 5.8. Classification of Yen-Guo Algorithm.

Domain	Uncompressed images
Perception-level	Low or None
Cryptosystem	XOR
Encryption Approach	Naïve, Chaos-based
Format Compliant	--
Bitrate	Constant

5.7 IMAGE ENCRYPTION BY FRIDRICH

An important proposal for image encryption based on 2-D chaotic maps was published by Fridrich in 1997 [51]. Her proposal was an extension of the previous work by Pichler and Scharinger [52,53] who first introduced encryption schemes based on the Baker map.

5.7.1 Fridrich Algorithm

Jiri Fridrich proposed a five-step process for encrypting image data by means of a two-dimensional chaotic map, such as the Baker map [51]. Without loss of generality, assume that we are encrypting an uncompressed image with $N \times N$ pixels and L levels of gray.

In the first step, one chooses an appropriate two-dimensional chaotic bijection on the real unit set $[0,1]\times[0,1]$. The following map, called the Baker map, is recommended by Fridrich [51]:

$$B_b(x,y) = (2x,\ y/2) \quad \text{if } 0 \le x < 0.5$$

$$B_b(x,y) = (2x-1,\ y/2+0.5) \quad \text{if } 0.5 \le x \le 1$$

The second step in Fridrich Algorithm involves creating the generalized map. The Baker map can be generalized as follows. Divide the unit range $[0,1]\times[0,1]$ into k vertical rectangles $[F_{i-1},F_i)\times[0,1)$, so that $F_0 = 0$ and $F_i = p_1 + \ldots + p_i$, $1 \le i \le k$. This can be described by the following formula:

$$B_g(x,y) = ((x - F_i)/p_i,\ p_i y + F_i) \quad \text{for } (x,\ y) \in [F_i,\ F_i + p_i)\times[0,1).$$

The third step is the discretization of the generalized chaotic map. For the Baker map, one defines a sequence of k integers, n_i $(1 \le i \le k)$ such that $n_1 + \ldots + n_k = N$. If we define $N_0 = 0$ and $N_i = n_1 + \ldots + n_k$, where n_i divides N, then the pixel $(r,\ s)$ with $N_i \le r < N_i + n_i$. and $1 \le s < N$ is mapped under the discretized Baker map as follows:

$$B_d(r,\ s) = (q_i \times (r - N_i) + (s \bmod q_i),\ (s - (s \bmod q_i)) \ /\ q_i + N_i),$$

where $q_i = N \ /\ n_i$.

When applied to an image, this map permutes the pixel positions. Figure 5.16 shows the results of applying the above transformation to "Lena". In the next step, the discretized map is extended to 3-D in order to transform not just the pixel positions but the pixel values (gray levels) as well. Fridrich defines the following method that could be applied to any 2-D map. Let $g_{(i,\ j)}$ be the grey level value at pixel position $(i,\ j)$. We transform (encrypt) this pixel using the following three-dimensional map B:

$$B((i, j), g_{(i,j)}) = (B_d(i, j), h(i, j, g_{(i,j)})),$$

where $h(i, j, g_{(i,j)}) = g_{(i,j)} + h'(i, j) \bmod L$, and where h' is arbitrary function on i and j. In [], Fridrich chose h to be:

$$h(i, j, g_{(i,j)}) = g_{(i,j)} \oplus (i \times j \bmod L).$$

Figure 5.17 shows applying this three-dimensional transformation to "Lena". This obviously introduces better perceptual distortion than the two-dimensional transformation shown in Figure 5.16.

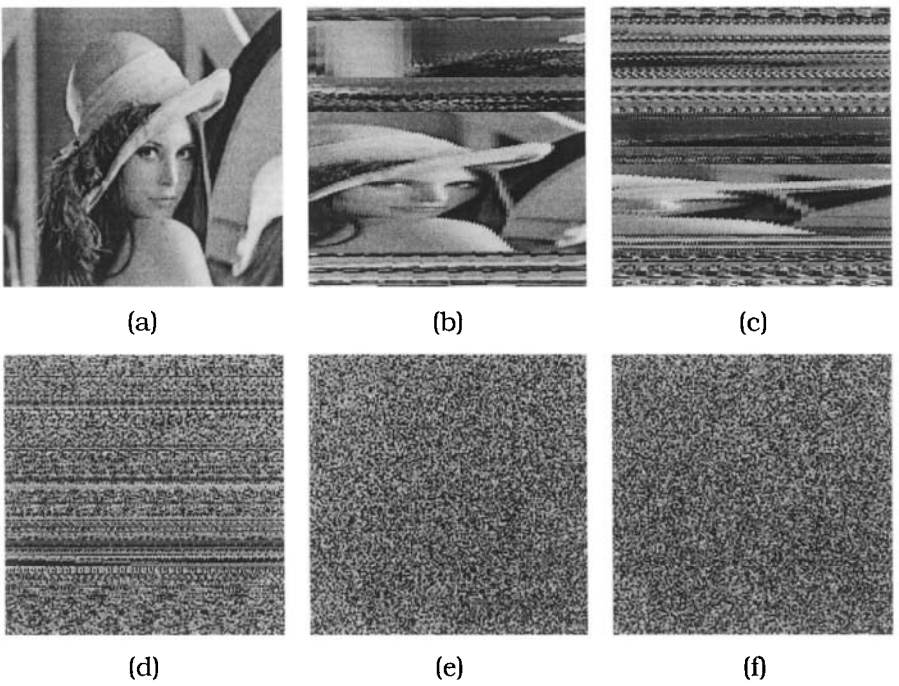


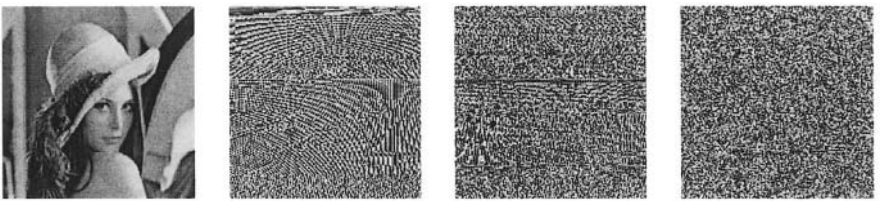
Figure 5.16. Encrypting 160×160 8-bit grayscale "Lena" using a 2-D discretized Baker map with the secret key (k=7, n_1=8, n_2=32, n_3=20, n_4=80, n_5=10, n_6=8, n_7=2), and a various number of rounds: (a) Original image, (b) 1 round of encryption, (c) 2 rounds of encryption, (d) 4 rounds of encryption, (e) 8 rounds of encryption, and (f) 16 rounds of encryption.

As a final step, the Fridrich Algorithm introduces a diffusion step. The author suggests applying a simple non-linear feedback shift

register. Set $g_{(0,-1)}$ to some initial condition (an integer between 0 and $L - 1$), and change the gray levels of the encrypted image from $g_{(i,j)}$ to $g^*_{(i,j)}$ using the following transform:

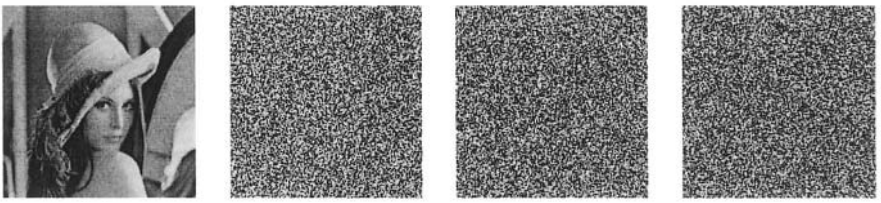
$$g^*_{(i,j)} = g_{(i,j)} + G(g^*_{(i,j-1)}) \bmod L,$$

where G is some arbitrary function of the gray level, such as a fixed random permutation [51]. Figure 5.18 shown the full Fridrich Algorithm applied to "Lena".



(a) (b) (c) (d)

Figure 5.17. Encrypting 160×160 8-bit grayscale "Lena" using a 3-D discretized Baker map (without the feedback step) and with the secret key (k=7, n_1=8, n_2=32, n_3=20, n_4=80, n_5=10, n_6=8, n_7=2), and a various number of rounds: (a) Original image, (b) 1 round of encryption, (c) 2 rounds of encryption, (d) 4 rounds of encryption.



(a) (b) (c) (d)

Figure 5.18. Encrypting 160×160 8-bit grayscale "Lena" using the full Fridrich Algorithm- a 3-D discretized Baker map with the feedback step, and with the secret key (k=7, n_1=8, n_2=32, n_3=20, n_4=80, n_5=10, n_6=8, n_7=2, $g^*_{(0,-1)}$=32), and a various number of rounds: (a) Original image, (b) 1 round of encryption, (c) 2 rounds of encryption, (d) 4 rounds of encryption.

Note that in this scheme, the secret key K is a sequence of initial integers $K = \{C, n_1, \ldots, n_k, g_{(0,-1)}\}$, where C is the number of iterations of

the Baker map. The decryption process consists of applying the inverse of these transformations to the pixels of the ciphertext image but in the reverse order (starting from step 5). Notice that all of the above transformations are invertible.

The Fridrich Algorithm seems to be a competent chaotic map version of a product cipher. In [54], it was argued that there are some weak keys in Fridrich's encryption scheme. The problem is caused by the short recurrent-period effect of the discretized version of the Baker map. Namely, there are keys for which the recurrent-period of the chaotic permutation is only 4 [54]. The scheme was further enhanced in [55] to overcome this problem. Namely, in [55] Fridrich Algorithm is modified to include an additional permutation that shifts the pixels in each image row.

Table 5.9. Classification of Fridrich Algorithm.

Domain	Uncompressed images
Perception-level	Low or None
Cryptosystem	Product cipher
Encryption Approach	Naïve, Chaos-based
Format Compliant	--
Bitrate	Constant

5.8 IMAGE ENCRYPTION BY CHEN, MAO AND CHUI

In [56], Chen, Mao, and Chui suggested a new approach for fast and secure image encryption based on a three-dimensional chaotic map. To permute (diffuse) a two-dimensional image, a discrete three-dimensional chaotic map is designed and then used to shuffle the positions of pixels. To substitute (confuse) the relationship between the ciphertext image and the plaintext image, a substitution process among pixels is performed using another chaotic map. The authors suggest using the discretized Arnold's cat map that is extended to a three-dimensional version, called *3D cat map.*

5.8.1 3D Cat Map

Arnold's cat map is a well-known two-dimensional invertible chaotic map [56] that is defined as follows:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \times \begin{bmatrix} x_n \\ y_n \end{bmatrix} \pmod 1,$$

where the standard notation "x (mod 1)" represents the fractional part of a real number x.

The two-dimensional cat map could be generalized by establishing the following two control parameters, a and b:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & a \\ b & ab+1 \end{bmatrix} \times \begin{bmatrix} x_n \\ y_n \end{bmatrix} \pmod 1.$$

The generalized two-dimensional cat map from above can be extended to three-dimensions by applying it on each of the three planes (X-Y, X-Z, and Y-Z). When combined together, this results in the following three-dimensional extension:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{bmatrix} = A \times \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} \pmod 1,$$

where

$$A = \begin{bmatrix} 1+a_x a_z b_y & a_z & a_y + a_x a_z + a_x a_y a_z b_y \\ b_z + a_x b_y + a_x a_z b_y b_z & a_z b_z + 1 & a_y a_z + a_x a_y a_z b_y b_z + a_x a_z b_z + a_x a_y b_y + a_x \\ a_x b_x b_y + b_y & b_x & a_x a_y b_x b_y + a_x b_x + a_y b_y + 1 \end{bmatrix},$$

and a_x, b_x, a_y, b_y, a_z, and b_z, are all positive integer parameters.

5.8.2 Chen-Mao-Chui Algorithm

The image encryption algorithm by Chen, Mao and Chui consists of five stages: key generation, 3-D pile-up, permutation, substitution, 2-D spread-out.

The first stage, known as the *key generation*, takes a 128-bit key k and accordingly produces the algorithm parameters. The binary sequence k is divided into eight 16-bit segments denoted by k_{ax}, k_{bx}, k_{ay}, k_{by}, k_{az}, k_{bz}, $k_{a\prime}$, and $k_{b\prime}$. Segments k_{ax}, k_{bx}, k_{ay}, k_{by}, k_{az}, and k_{bz} are used to obtain three-dimensional cat map parameters a_x, b_x, a_y, b_y, a_z, and b_z used in the permutation stage, while $k_{a\prime}$ and $k_{b\prime}$ are used to obtain the two parameters for a one-dimensional Logistic map that is later used in the substitution stage. To obtain the 3-D cat map parameters, the algorithm uses Chen's chaotic system [57,58]. Chen's system is defined as follows:

$$\begin{cases} x_{n+1} = \alpha(y_n - x_n) \\ y_{n+1} = (\gamma - \alpha)x_n - x_n z_n + \gamma y_n \\ z_{n+1} = x_n y_n - \beta z_n, \end{cases}$$

where a, b and c are the control parameters. When $\alpha = 35$, $\beta = 3$, and $\gamma \in [20, 28.4]$, then the Chen's system behaves chaotically. According to the Chen-Mao-Chui Algorithm, we first compute Chen's parameter γ by using the following formula:

$$\gamma_i = K_{a_i} \times 8.4 + 20,$$

where $K_{a\prime}$ represents normalized decimal representation of $k_{a\prime}$, $i \in \{x, y, z, \prime\}$. The initial values $x_{0\prime}$, $y_{0\prime}$, $z_{0\prime}$, $i \in \{x, y, z, \prime\}$, of Chen's system are derived by the following formula:

$$\begin{cases} x_{0_i} = K_{b_i} \times 80 - 40 \\ y_{0_i} = K_{a_i} \times 80 - 40 \\ z_{0_i} = K_{b_i} \times 60. \end{cases}$$

By setting Chen's parameters α_i and β_i to constant values 35 and 3, respectively, we iterate the system 100 and 200 times to obtain values z_{100i} and z_{200i}. Next, the following equations are used to

generate the final parameters a_i and b_i, $i \in \{x, y, z\}$, for the 3-D cat map:

$$a_i = \lfloor (z_{100_i} / 60 \times N) + 0.5 \rfloor$$
$$b_i = \lfloor (z_{200_i} / 60 \times N) + 0.5 \rfloor,$$

where N represents the side length of a cube created by the 3-D data pile-up. The parameters a_l and b_l for the Logistic map are calculated as follows:

$$a_l = z_{100_l} / 60$$
$$b_l = \lfloor (z_{200_l} / 60 \times 255) + 0.5 \rfloor.$$

In the second stage, known as the *3-D pile-up*, the pixel values within the two-dimensional image are piled up into several three-dimensional cubes. An $W \times H$ image is piled up into the cubes of size $N_1 \times N_1 \times N_1$, $N_2 \times N_2 \times N_2$, ..., $N_i \times N_i \times N_i$, respectively, such that the following holds:

$$W \times H = N_1^3 + N_2^3 + \cdots + N_i^3 + R,$$

where $N_i \in \{2, 3, \ldots, N\}$, N is the size of maximum allowable cube, and the reminder $R \in \{0, 1, \ldots, 7\}$.

The third stage, or *permutation* stage, consists of applying the three-dimensional discrete cat map with control parameters a_x, b_x, a_y, b_y, a_z, and b_z to each of the image cubes, resulting in a permuted image.

In the fourth stage, known as the *substitution* stage, the image values are substituted with other values to further enhance the encryption effect. The substitution is achieved by using the following Logistic map:

$$x_n = 4x_{n-1}(1 - x_{n-1}),$$

where $x_0 = a_l$. It is desired that all of the values in the iterated sequence be in the interval (0.2, 0.8). If the obtained value x_i is not in the interval (0.2, 0.8), repeat the iteration again. Additionally, if x_i is exactly 0.5, a perturbation should be applied to avoid the iteration

trap. The substitution of the i-th pixel I_i by C_i is performed in the following way. Let ϕ_i denote the discretized and scaled version of x_i. Then, apply the following substitution:

$$C_i = \phi_i \oplus \left((I_i + \phi_i) \bmod N \right) \oplus C_{i-1},$$

where $C_0 = b_i$, and N is the number of colors (N = 256 for grayscale images).

The final stage is the *2-D spread-out*. Here the image is put back from the three-dimensional representation and arrangement to the usual two-dimensional $W \times H$ arrangement.

The decryption is straight forward, and consists of applying the appropriate inverse transformations. The effects of Chen-Mao-Chui Algorithm are shown in Figure 5.19.

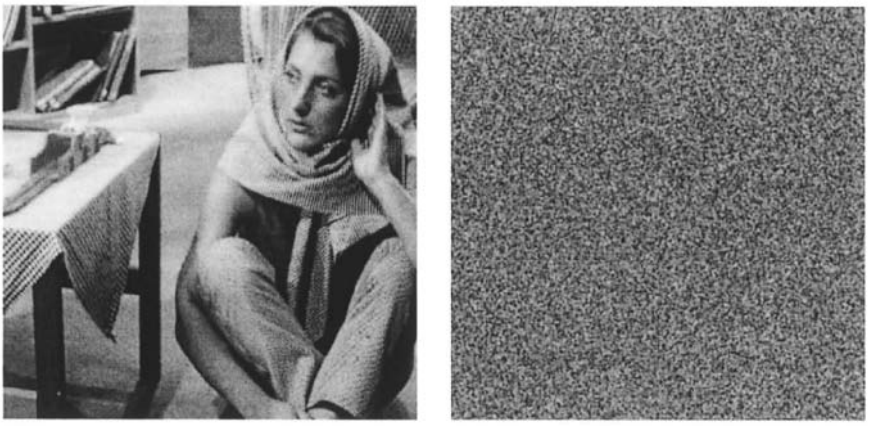


Figure 5.19. Encrypting 512×512 image "Barb" with Chen-Mao-Chui Algorithm: (a) the original image, and (b) the encrypted image obtained using the 16-character key "1234567890123456" (128 bits) (© 2004 IEEE).

According to the experimental results from [56], Chen-Mao-Chui Algorithm possesses the strong randomness properties needed for all strong cryptosystems. Experiments suggest that the system produces the ciphertext that has very weak or no correlation with the original image. This important property is needed to withstand all kinds of statistical attacks. In addition, the scheme has strong

avalanche property, which is essential to avoid differential attacks. Perhaps a possible improvement of the system would be to replace the Logistic map with ICMIC map [45] or PWLCM map [46], which are known to have a better balance property.

Table 5.10. Classification of Chen-Mao-Chui Algorithm.

Domain	Uncompressed images
Perception-level	Low or None
Cryptosystem	Product cipher
Encryption Approach	Naïve, Chaos-based
Format Compliant	--
Bitrate	Constant

5.9 IMAGE ENCRYPTION BY VAN DROOGENBROECK

In [59], Marc Van Droogenbroeck defined a *generalized selective image encryption*. This concept extends the classical selective encryption scenario to include additional features. Van Droogenbroeck argued that the generalized selective approach should provide methods for the following four features:

- *Scalability (flexibility)* – a user should be able to select different security and perception levels. For example, this could include selecting different subsets of DCT coefficients for encryption.
- *Multiplicity* – two or more different users may want to encrypt common information. Without loss of generality, assume that two users A and B own a set of images C_A and C_B. If both users encrypt their set of images, then $C_A \cap C_B$ is doubly encrypted. According to [60,3] the coefficients that are encrypted twice or more are sensitive to attacks, so one should use *over-encryption* [59] to improve the security. In the case of two parties, doubly encryption of x, denoted by $E_{k1}(E_{k2}(x))$, is performed by applying the encryption function E twice, with different keys k_1 and k_2. In this notation, the over-encryption of x is denoted by $E_{k1}(D_{k2}(E_{k1}(x)))$, where D represents the decryption function. The general selective

encryption approach should allow utilizing over-encryption for multiple encryptions in a multi-user environment.

- *Spatial selectivity* – a user should be able to select different parts of the image to be heavily encrypted. The selected zones are marked in a binary map, called the *selection map*. This is useful in applications such as censoring, where only the censored part of the image is scrambled to the non-perceptible level. Figure 5.20 shows a locally encrypted image (spatial selectivity).
- *Self-sufficiency and format-compliance* – the decoder should be able to read the bitstream (format-compliance) and to retrieve the information about owners and their selection maps (self-sufficiency). Additional information that decoder should have at the decryption stage is which algorithm (conventional or not) was used for the encryption. All this information should be embedded in the bitstream. One could for example use an embedding scheme proposed by Fridrich, et al. in [61]. The embedding schemes, however, increase the size of the bitstream.

In [59], Van Droogenbroeck argued that the format compliance should be achieved by reassembling all of the data, encrypted or not, into a format compliant bitstream. That is, the encoder should create the compatible headers and other markers after the encrypted and unencrypted data is merged, in order to achieve compliance at the decoder side.

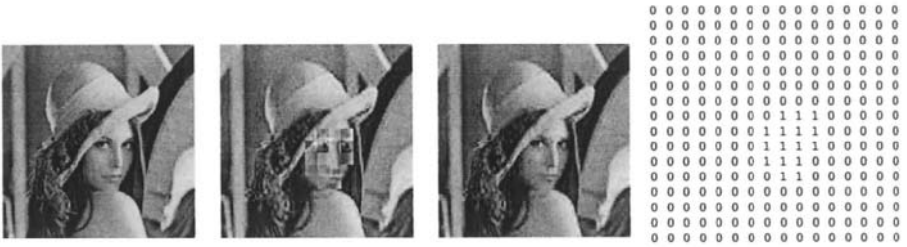


Figure 5.20. Van Droogenbroeck's spatial selectivity: (a) Original 128×128 "Lena", (b) locally encrypted "Lena" according to the selection map from (d) seen from the ordinary JPEG decoder, (c) correctly decrypted "Lena", (d) the selection map used for encryption.

Table 5.11. Classification of Van Droogenbroeck's Generalized Selective Encryption.

Domain	Compressed images
Perception-level	Variable
Cryptosystem	Any
Encryption Approach	Generalized Selective
Format Compliant	Yes
Bitrate	Variable

SUMMARY

Many algorithms specifically designed for encrypting digital images are proposed since mid-1990s. There are two major groups of image encryption algorithms: non-chaos selective methods, and chaos-based selective or non-selective methods. Most of these algorithms are designed for a specific image format (compressed or uncompressed), and some of them are even format compliant. There are schemes that offer a light encryption (degradation), while other offer stronger form of encryption. Some of the algorithms are even scalable and have different modes ranging from degradation to strong encryption. The user is expected to choose a method based on its properties, so that it best suits a given image security application.

Chapter 6

VIDEO ENCRYPTION ALGORITHMS

In this section, we discuss different research directions, which were taken in the area of video communication encryption. As discussed earlier, a naïve approach can be used to encrypt multimedia content. The whole multimedia stream is encrypted using a conventional cryptosystem. However, even the fastest modern symmetric schemes are computationally very expensive for many real-time video applications. Digital videos are very large files that usually require extremely large bitrates. Moreover, consecutive frames within a video are likely to be very similar, which requires ciphers with exceptional avalanche property. Finally, videos are likely to contain predictable frames such black introductory frames, MPAA ratings, FBI warnings, MGM roaring lion sequences, or other usual jingles. In that respect, ciphers that are vulnerable to known-plaintext attacks are not favorable.

6.1 SELECTIVE VIDEO ENCRYPTION

Just like in the case of images, the idea of selective encryption could be applied to encrypt only a portion of the compressed bit-stream to effectively improve the performance. A common approach in selective encryption is to integrate compression with encryption as shown in Figure 6.1. Note that in the case of digital videos, the data is rarely kept uncompressed due to the huge storage requirements that such data would impose. Hence, encryption algorithms designed for videos deal with the usual video compression formats from either MPEG family or H.26x family. All such formats have some common stages used to exploit spatial and temporal redundancies of the raw uncompressed sequences of frames. Thus, many of the algorithms presented in this chapter are applicable to multiple formats, regardless of the original intended design. Furthermore, some of the new video formats did not even exist at the time when some of these encryption algorithms were proposed.

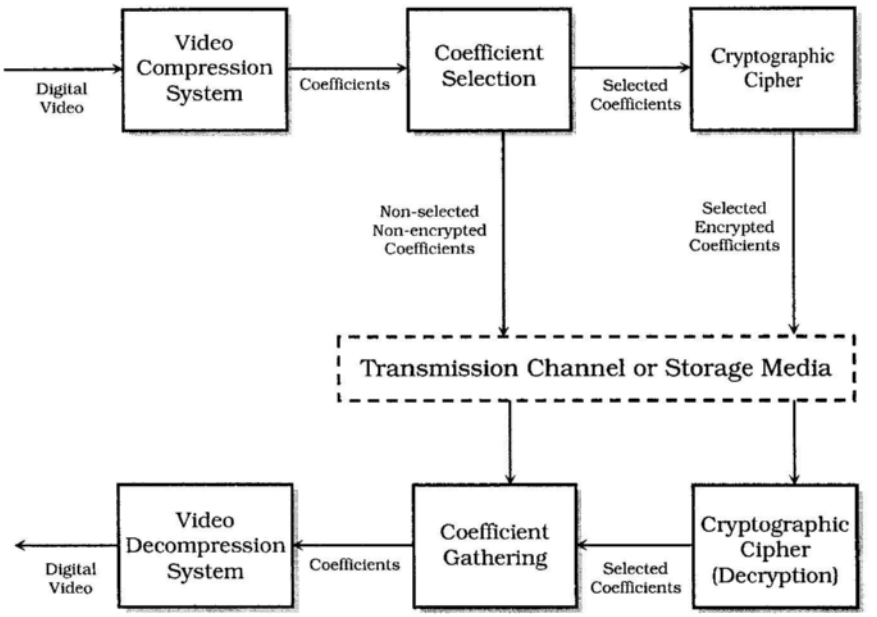


Figure 6.1. A common selective encryption scheme for compressed videos.

For that reason, the algorithms in this chapter are designed in close relation with a particular compression algorithm/format. For example, one of the popular approaches is to select only the most important coefficients from either final or intermediate steps of a compression process and encrypt only those coefficients. Less important coefficients are left unencrypted. Still, some schemes prefer to lightly encrypt (degrade) even these less important coefficients. In this manner, we define a *variable encryption* as an approach that applies different encryption schemes with different security strengths. Selective encryption can be treated as a special case of a variable encryption [62].

6.2 BASICS OF MPEG VIDEO COMPRESSION

Motion Picture Experts Group, an organization established in 1988, is responsible for creating a family of video compression standards called MPEG codecs (MPEG-1, MPEG-2, MPEG-4, MPEG-7, and MPEG-21). Most of the video encryption algorithms discussed in this chapter are designed for MPEG videos. It is assumed that the

reader is familiar with a codec from the MPEG family, so that we skip discussing the details. Figure 6.2 shows the basic stages of MPEG encoding algorithm. More on MPEG could be found in [63,64].

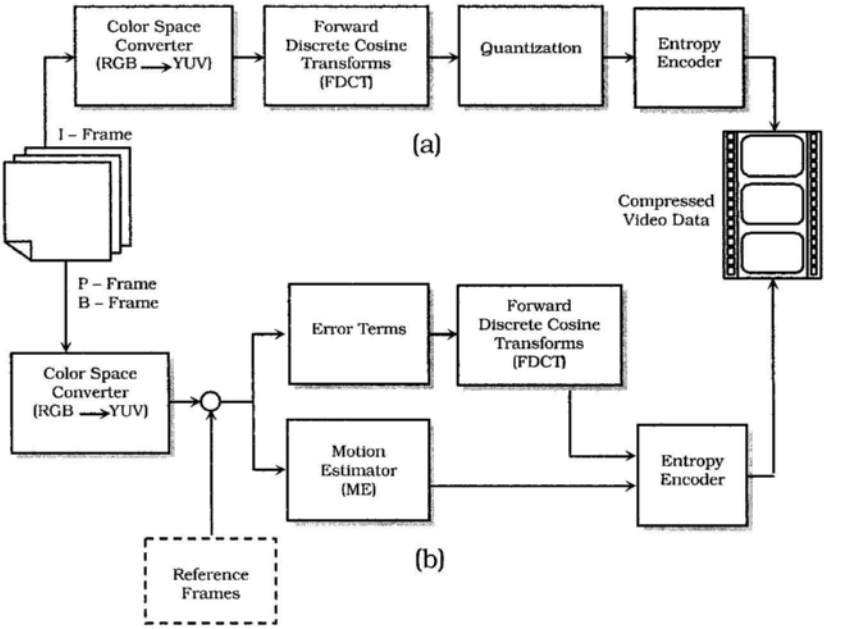


Figure 6.2. Stages of MPEG encoder: (a) encoding I-frame, and (b) encoding P- and B-frame.

In short, an MPEG codec exploits both spatial and temporal redundancies to achieve good compression of digital videos. Transform coding, such as Discrete Cosine Transforms (DCT), is utilized to exploit spatial redundancies, while motion estimation and compensation are used to exploit temporal redundancies. In this process, three types of frames are created. The I-frames are encoded to utilize spatial redundancies only (see Figure 6.2-a). In fact, the I-frames are encoded similarly to a JPEG image. On the other hand, P-frames and B-frames depend on one or more reference frames (see Figure 6.2-b).

6.3 VIDEO ENCRYPTION BY MEYER AND GADEGAST

In 1995, Meyer and Gadegast introduced the selective encryption method called *Secure MPEG*, or shortly *SECMPEG*, designed for the MPEG-1 video standard [65]. The SECMPEG can be naturally extended to later versions of MPEG codecs. Even though this scheme was later proved less secure than originally thought, with this work, Meyer and Gadegast are considered as one of the pioneers in the area of selective video encryption. The SECMPEG paper [65] and implementation by Meyer and Gadegast was one of the first important research initiatives for selective encryption of multimedia streams. These authors were among the first to realize the benefits of encrypting only selected bits in a video bitstream. Their experiments confirmed that the visual disruption was substantial when only encrypting the DC coefficients and the first 3-8 AC coefficients in the I-frame block of the MPEG-1.

6.3.1 Meyer-Gadegast Algorithm (SECMPEG)

The SECMPEG contains four different levels of security, with each level selecting more data to undergo the encryption stage. The authors utilized Data Encryption Standard (DES) cryptosystem. Since DES is a symmetric key cryptosystem, it could only be used to achieve confidentiality. Meyer and Gadegast targeted solving the problem of data integrity as well. For that reason, the Cyclic-Redundancy-Check (CRC) was incorporated as a low-level solution to the integrity.

The MPEG-1 encoded bitstream consists of the following six layers, out of which the first four layers are not entropy encoded and contain headers:

1. Video Sequence Layer
2. Group of Pictures (GOP) Layer
3. Picture Layer
4. Slice Layer
5. Macroblock Layer
6. Block Layer

These hierarchical structures are shown in Figure 6.3. The top layer is the video sequence, which consists of several *Groups of Pictures* (GOPs). The header of Video Sequence Layer contains the

information on horizontal and vertical frame size, pixel aspect ratio, picture rate, bitrate, and buffer size. The GOP Layer contains a particular sequence of I-frames, P-frames, and B-frames. Each frame constitutes a Picture Layer, whose header contains information about the temporal reference (the order in which the frames should be displayed), the picture type (I, P, or B), and the pixel accuracy. A frame is further divided into *slices*, each consisting of several blocks that are convenient for error concealment when transmission error occurs. *Macroblock* is a 16×16 pixel area for the luminance, and a 8×8 pixel area for the chrominance, and it represents the basic unit for motion estimation. It also constitutes the Macroblock Layer. Finally, *block* is an 8×8 pixel area of a frame that represents the basic unit for DCT.

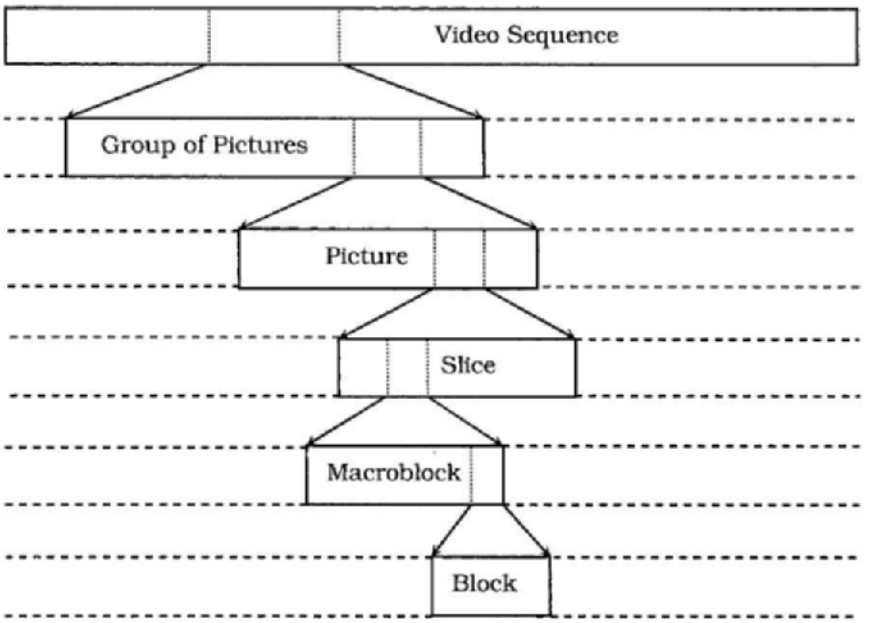


Figure 6.3. Layered hierarchy of an MPEG-1 bitstream.

The algorithm proposed by Meyer and Gadegast takes an MPEG-1 bitstream as the input and performs one of the four different selective encryption transformations. These transformations are denoted by Level i, $1 \leq i \leq 4$, and that is based on their actual

security level. For instance, at Level 1, the algorithm's security is lowest, and so on.

At Level 1, SECMPEG encrypts only the headers of the highest four layers (Sequence Layer, GOP Layer, Picture Layer, and Slice Layer). At Level 2, in addition to encrypting the headers of the first four layers, SECMPEG encrypts the first macroblock after each slice header. By doing so, all offsets are encrypted, which makes it difficult to locate the relevant data. At Level 3, the algorithm encrypts all I-frames and all intracoded macroblocks. Finally, at Level 4, SECMPEG encrypts the entire MPEG-1 sequence (the naive approach). On the other end, a special SECMPEG decoder is needed to recover the original MPEG-1 video stream. During decryption process, SECMPEG also performs one of the four different methods for the data integrity. As mentioned earlier, these methods are based on CRC. In the first method, a 16-bit CRC is performed on all header data. In the second method, a 16-bit CRC is performed on all header data and all frames. A 32-bit CRC is performed on all header data in the third method. Finally, in the fourth method, a 32-bit CRC is performed on all header data and all frames. All CRC-checks are very quick, and the authors suggested using the fourth method for integrity.

The selective encryption of SECMPEG at Levels 1, 2, and 3 has weaknesses [68,35]. At Levels 1 and 2, brute force can be utilized to recover the scrambled header information [65]. In addition, SECMPEG at Level 3 is also shown to be insecure. Namely, even though single P-frame or B-frame on its own carries almost no information without the corresponding I-frame, a series of P-frames or B-frames can tell a lot if their base I-frames are correlated. The experiments by Agi and Gong showed that encrypting I-frames only might leave visible I-blocks in other frames [68]. The same authors then propose a few trade-off improvements: increasing the frequency of the I-frames and/or encrypting all P-frames and B-frames. These improvements decrease speed, and further disrupt the compression ratio.

Since SECMPEG introduces changes to the MPEG-1 format, a special encoder and decoder is needed to handle SECMPEG streams. In other words, Meyer-Gadegast Algorithm is not format compliant. In respect to this, the effects of compression are diminished since the file size increases when its portions are encrypted using DES

block cipher, and new headers introduced. At Levels 1 and 2, the algorithm encrypts very small portion of the bitstream, but at the very high expense of security. At Level 3, the amount of selected bits for encryption depends on a GOP structure of the MPEG-1 video: the more I-frames and intracoded macroblocks, the higher percentage of the bitstream undergoes encryption. Unfortunately, even though the algorithm performs fast when the stream contains a small amount I-frames in the GOP, its security is low due to reconstruction procedures based on many related P-frames and B-frames from the GOP. In addition, the proposed CRC routines are fast, however, to ensure data integrity on a cryptographic level, one must use conventional hash functions and public key cryptography whose speed is much lower.

Table 6.1. Classification of Meyer-Gadegast Algorithm.

Domain	MPEG-1 videos
Perception-level	Variable
Cryptosystem	Conventional
Encryption Approach	Selective
Format Compliant	No
Bitrate	Near-constant

6.4 VIDEO ENCRYPTION BY MAPLES AND SPANOS

Another important research achievement on selective encryption of MPEG video stream appeared in 1995 by Maples and Spanos [66,67], and it introduced a secure MPEG video mechanism called Aegis. Aegis was initially designed for MPEG-1 and MPEG-2 video standards, however the same encryption principles can be easily extended to other MPEG standards.

6.4.1 Maples-Spanos Algorithm (Aegis)

The Maples-Spanos Algorithm (Aegis) encrypts I-frames of all MPEG groups of frames in an MPEG video stream, while B- and P-frames are left unencrypted. In addition, Aegis also encrypts the MPEG video sequence header, which contains all of the decoding initialization parameters that include the picture width, height, frame rate, bit rate, buffer size, etc. Finally, this method also encrypts the ISO end code; i.e., the last 32 bits of the stream. As a

result, the bit-stream's MPEG identity is further concealed. The Aegis' behind-the-scenes encryption engine chosen by Maples and Spanos was DES, however, without loss of generality, any conventional cipher could be utilized for that matter. In a way, Aegis is very similar to the level 3 of SECMPEG by Meyer and Gadegast, and the security weaknesses that are applicable to SECMPEG are carried over to Aegis.

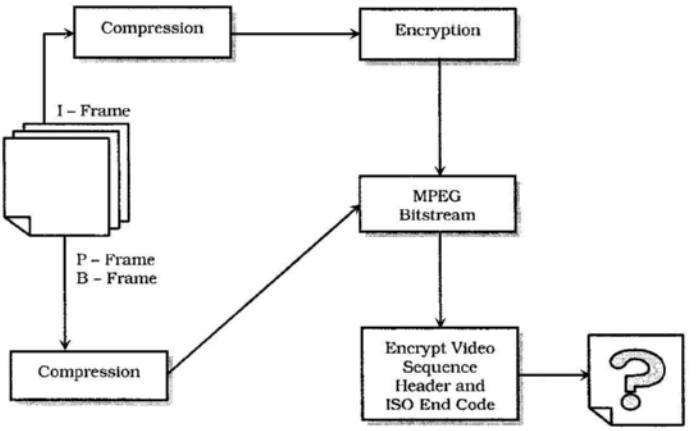


Figure 6.4. Stages of Maples-Spanos Algorithm (Aegis) for MPEG Video Encryption.

As discussed earlier, Agi and Gong criticized encrypting only I-frames by Aegis and SECMPEG, in view of the fact that their experiments showed that it is possible to discern some aspects from the scene from the decoded P- and B-frames [68]. For example, they were able to conclude that the talking head was present at the scene. Alattar and Al-Regib presented the similar argument regarding the security of SECMPEG in [70]. For that reason, Maples-Spanos Algorithm is not particularly good for applications where the security is one of the top priorities. Aegis requires special kind of encoder and decoder, since the bitstream is not format-compliant.

Aegis is also considered as one of the pioneering research accomplishments in the area of selective video encryption. The experimental results for the selective encryption published by

Maples and Spanos in [66,67] confirmed the enormous gain in speed over the naive approach, by applying the selective encryption.

Table 6.2. Classification of Maples-Spanos Algorithm.

Domain	MPEG videos
Perception-level	Low
Cryptosystem	Conventional
Encryption Approach	Selective
Format Compliant	No
Bitrate	Near-constant

6.5 VIDEO ENCRYPTION BY QIAO AND NAHRSTEDT

After performing extensive statistical analysis of MPEG video streams and discovering weaknesses in previously proposed approaches that include Meyer-Gadegast Algorithm, Maples-Spanos Algorithm and Tang Algorithm, Lintian Qiao and Klara Nahrstedt proposed their own selective video encryption approach. In their method [69], which they simply called a video encryption algorithm (VEA), Qiao and Nahrstedt proposed a neat way to achieve the security level similar to that of a naïve approach by encrypting only half of the bitstream.

6.5.1 Qiao-Nahrstedt Algorithm (VEA)

The Qiao-Nahrstedt Algorithm (VEA) [69] is constructed with the goal to exploit the statistical properties of the MPEG video standard. The algorithm consists of the following four steps:

- *Step 1:* Let the $2n$ byte sequence, denoted by $a_1a_2...a_{2n}$, represent the chunk of an I-frame
- *Step 2:* Create two lists, one with odd indexed bytes $a_1a_3...a_{2n-1}$, and the other with even indexed bytes $a_2a_4...a_{2n}$.
- *Step 3:* Xor the two lists into an n-byte sequence denoted with $c_1c_2...c_n$
- *Step 4:* Apply the chosen symmetric cryptosystem E (for example DES or AES) with the secret key *KeyE* on either odd list or even list, and thus create the ciphertext sequence

$c_1c_2...c_n E_{KeyE}(a_1a_3...a_{2n-1})$ or $c_1c_2...c_nE_{KeyE}(a_2a_4...a_{2n})$ respectively.[4]

Clearly, the decryption mechanism at the other end of the communication channel consists of two easy steps:

1. Apply the symmetric cryptosystem E with the appropriate key to the second half of the ciphertext to obtain the first half of the original sequence.
2. XOR the result with the first half of the ciphertext to obtain the other half of the original sequence.

Even though the security of the system is based on the security of the chosen underlying cryptosystem, if an adversary obtains either even or odd list, the system is completely broken. Thus, the authors suggest an even more secure approach. Instead of dividing the list into the even and odd lists, the random $2n$-bit key, called *KeyM*, is generated and used to split the $2n$-byte chunk of MPEG stream into two lists. The method of partitioning the $2n$-byte MPEG sequence into two subsequences is as follows: if i^{th} bit in *KeyM* is 1 then the i^{th} byte of the MPEG sequence goes into the first subsequence, and similarly, if i^{th} bit in *KeyM* is 0 then the i^{th} byte of the MPEG sequence goes into the second subsequence. It is required that *KeyM* is a binary sequence with exactly n ones and n zeros, in order to ensure that the two lists are of the equal length. The suggested values for n are 64 or 128. By applying this approach, the attacker will have a very hard time ordering the resulting sequence of bytes in the case that the encrypted subsequence is somehow cracked, assuming that the binary sequence *KeyM* remained secret. This means that *KeyM* must be securely transmitted to the receiver before the decryption can take place.

Another consideration is that a possible pattern repetition in the $2n$ stream chunk represents a significant security hole in the method proposed be Qiao and Nahrstedt. According to the statistical analysis of MPEG stream sequences, it was observed that the non-repeating patterns have a lifetime over only one 1/16 chunk [69]. Therefore, to obtain the non-repeating pattern with a length of 1/2 frame, which is consisted of 8 such chunks, it is sufficient to shuffle

---

[4] The authors' choice was to encrypt the even sequence creating the ciphertext $c_1c_2...c_nE_{KeyE}(a_2a_4...a_{2n})$.

only these 8 chunks. However, each of these chunks must be shuffled by using different keys, which are here denoted by Key_1, Key_2, ..., Key_8. The authors suggest using the permutations of degree 32 (even though the degree 24 was shown to be sufficient), so that each Key_i is stored as a random permutation of degree 32. Hence, the length of a Key_i is 20 bytes, resulting in the total length of 160 bytes for Key_1, Key_2, ..., Key_8. Once the keys are generated, Key_1 permutation is applied to the first 32 bytes of even list, Key_2 permutation is applied to the next 32 bytes, and so on, repeating this process after the 8th key. This procedure will ensure the non-repetition in the MPEG stream, thus significantly decreasing the vulnerability of the system to the repetitive pattern attacks. Furthermore, the increase in the bit-stream size caused by including these extra keys is at a negligible level.

Finally, Qiao and Nahrstedt proposed the use of one more additional key, denoted by *KeyF*. The experiments have shown that the pattern of selecting the even list and the odd list should be changed for every frame in order to ensure the anticipated security level [69]. For this reason, a 64-bit key *KeyF* is randomly generated for every single frame in the MPEG sequence. However, this key is subject to the constraint that every 4 consecutive bits represent a unique number from 0 to 15; i.e., so that any *KeyF* represents a permutation of degree 16. To get the new keys for each frame, we permute *KeyM* and *Key*i's repeatedly using the permutation defined by *KeyF* assigned to that frame.

In order to securely transmit these additional keys (*KeyM*, *Key*i's and *KeyF*'s), they must be encrypted using the encryption function E with *KeyE*. As a final point, the secret key *KeyE* has to be securely transmitted to the decoder's side (the receiver), possibly by using public key cryptography, or by using a separate secure channel. Alternatively, a separate secure channel can be used for the transmission of keys *KeyM* and *Key*i's. The diagram of the proposed algorithm is shown in Figure 6.5.

The experiments that were originally performed in 1997 were mainly using DES and IDEA cryptosystems; however, a newer choice for the underlying cryptosystem E would probably be the recently standardized AES cryptosystem (Rijndael). The security of the method proposed by Qiao and Nahrstedt is very close to the security of encryption scheme E that is internally used. The speed of this

algorithm is roughly a twice the speed of naïve algorithm, but that is arguably still the large amount of computation for high quality real-time video applications that have high bit rates.

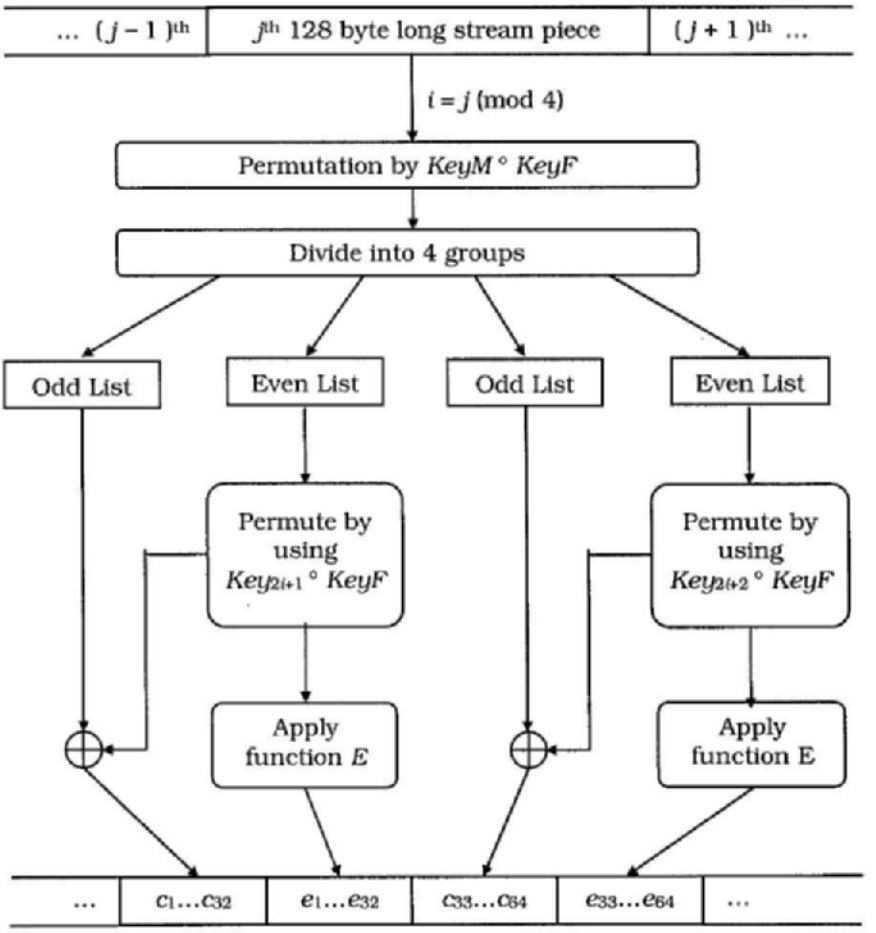


Figure 6.5. Selective Video Encryption Algorithm by Qiao and Nahrstedt.

Table 6.3. Classification of Qiao-Nahrstedt Algorithm.

Domain	MPEG videos
Perception-level	Low
Cryptosystem	Conventional
Encryption Approach	Selective
Format Compliant	No
Bitrate	Near-constant

6.6 VIDEO ENCRYPTION BY SHI, WANG AND BHARGAVA

In [38], Shi Wang and Bhargava classified their previous work [36,37] into four different video encryption algorithms: Algorithm I, Algorithm II (VEA), Algorithm III (MVEA), and Algorithm IV (RVEA). We have already discussed the first algorithm (in the previous chapter), which is applicable to both JPEG images and the I-frames of MPEG videos. However, Algorithms II, III and IV are only applicable to MPEG videos. These algorithms apply selective encryption in different stages of the MPEG encoder.

6.6.1 Shi-Wang-Bhargava Algorithm II (VEA)

The Algorithm II (VEA) uses the following selective encryption observation: it is sufficient to encrypt only the sign bits of the DCT coefficients in an MPEG video. The Algorithm II simply xors the sign bits of the DCT coefficients with a secret m-bit binary key $k = k_1k_2...k_m$ (see Figure 6.6). The effect of this scheme is that the encryption function randomly changes the sign bits of the DCT coefficients depending on the corresponding bit value of k. Let us denote the sign bits of the DCT coefficients in an MPEG stream that belong to the same GOP (Group Of Pictures) by $s_1s_2...s_n$. Then, a bit s_i will not be changed if the value of the key bit $k_{i \pmod m}$ is 0, and it will be flipped if the value of the key bit $k_{i \pmod m}$ is 1. The next GOP would simply reuse the secret key, which serves for the resynchronization purposes. The ability to resynchronize the video stream is important in the case of an unreliable network, transmission errors, and VCR-like functionality such as fast forwarding or rewinding.

The security of Algorithm II depends on the length of the key. The authors encourage the use of a long binary key; however, too long a key may be infeasible and impractical. On the other hand, if the key is short, the system can easily be broken. If the key is as long as the video stream and it is unique and used only once that would correspond to Vernam cipher (also referred to the one-time pad), which is known to be absolutely secure. However, this is highly impractical for mass applications such as VOD (Video on Demand) and similar. On the other hand, if the key is too short, the whole method simply becomes the Vigenére-like cipher, a well-studied classical cipher for which there are many attacks developed (for example, the Kasiski analysis) [39]. In addition, Vigenére is highly susceptible to the known-plaintext attack (one can literally read off the secret key).

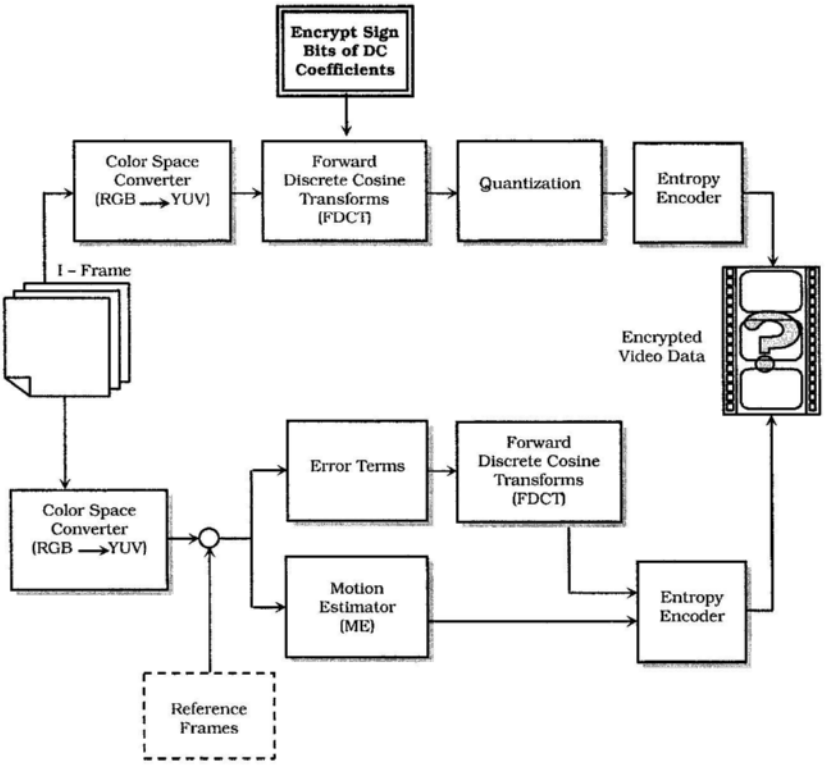
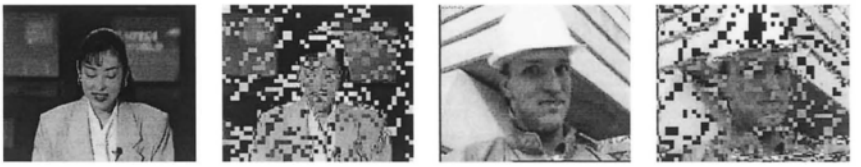


Figure 6.6. An MPEG video compression with Shi-Wang-Bhargava Algorithm II (VEA) encryption.

Authors in [38] suggest the use of the pseudo-random generator that generates a stream of pseudo-random bits to be the key k of arbitrary length. The trouble with this approach is the security, randomness, and speed of this P-RNG (Pseudo-Random Number Generator). The additional issues include the synchronization of the P-RNG and the secure transmission of the seed (the secret key) for the P-RNG.



(a) (b) (c) (d)

Figure 6.7. The leftmost two images show low-motion sequence "Akiyo" encrypted with Shi-Wang-Bhargava Algorithm II (VEA) and decoded with: (a) the correct key, and (b) the incorrect key. The rightmost two images show high-motion sequence "Foreman" encrypted with Shi-Wang-Bhargava Algorithm II (VEA) and decoded with: (c) the correct key, and (d) the incorrect key.

The Shi-Wang-Bhargava Algorithm II is fully format compliant and ordinary decoders can preview the encrypted MPEG video. However, the output is perceptually corrupted, and is fairly degraded. However, since many blocks were unchanged, the viewer can still see parts of the image. Figure 6.7 shows the effects of Shi-Wang-Bhargava Algorithm II applied on a low-motion sequence "Akiyo" and a high-motion sequence "Foreman".

Table 6.4. Classification of Shi-Wang-Bhargava Algorithm II

Domain	MPEG videos
Perception-level	Medium
Cryptosystem	Fixed key (Vigenére-like)
Encryption Approach	Selective
Format Compliant	Yes
Bitrate	Near-constant

6.6.2 Shi-Wang-Bhargava Algorithm III (MVEA)

The Algorithm III (MVEA) is an improvement to the Algorithm II (VEA) described above. It includes the following additions: the sign bits of differential values of motion vectors in P- and B-frames can also be randomly changed (see Figure 6.8). This type of improvement makes the video playback more random and more non-viewable. When the sign bits of differential values of motion vectors are changed, the directions of motion vectors change as well. In addition, the magnitudes of motion vectors change, making the whole video very chaotic. The authors found that the encryption of sign bits of motion vectors makes the encryption of sign bits of DCT coefficients in B- and P-frames unnecessary.

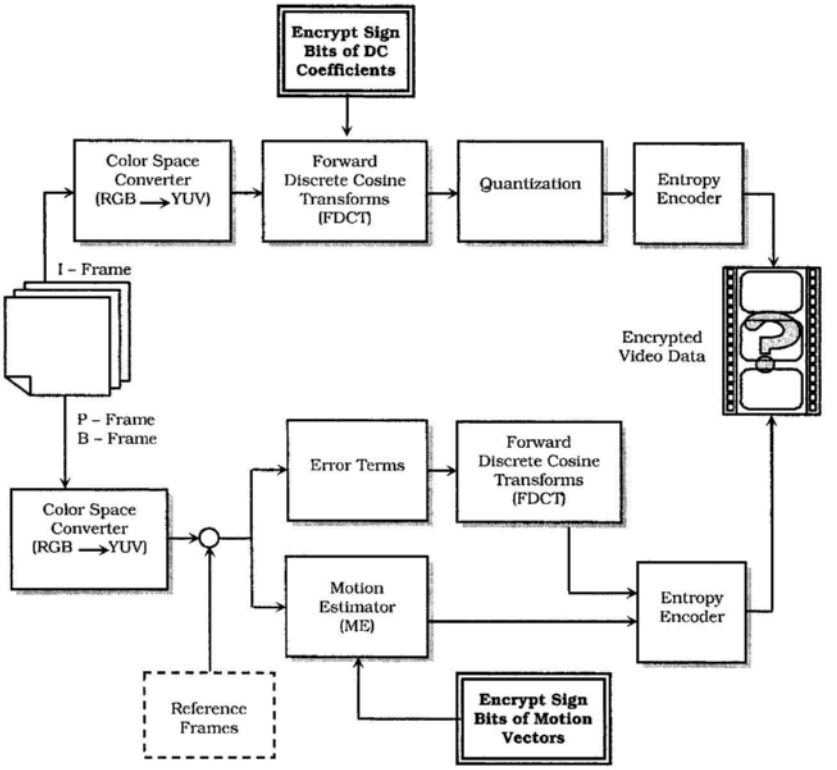


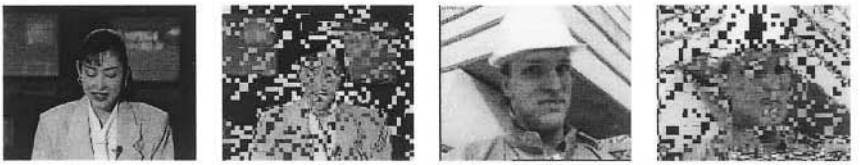
Figure 6.8. An MPEG video compression with Shi-Wang-Bhargava Algorithm III (MVEA) encryption.

Furthermore, the original Algorithm III (MVEA) was designed to encrypt only the sign bits of DC coefficients in the I-frames of MPEG video sequence, while leaving the AC coefficients unencrypted. This significantly reduces the computational overhead, but it opens up new security risks. Namely, because the DC coefficient and the sum of all AC coefficients within the block are related, an adversary may use the unencrypted AC coefficients to derive the unknown (encrypted) DC coefficients [37,38]. For that reason, the authors recommend encrypting all DCT coefficients in the I-frames for applications that need higher level of security.

Just like the Algorithm II (VEA), algorithm III (MVEA) relies on the secret m-bit key k. Also, the resynchronization is done at the beginning of a GOP. Unfortunately, the fundamental security issues and problems that are applicable to VEA are also applicable to MVEA. In addition, Shi-Wang-Bhargava Algorithm III is also fully format compliant, which means that we can view the encrypted videos with an ordinary MPEG decoder.

The effects of encrypting the motion vectors are more perceivable in the case of the high-motion video sequences. In the case of a low-motion video, the visual distortions produced by VEA and MVEA are perceptually very similar. Figure 6.9 shows the results of encrypting only the sign bits of motion vectors for both low-motion "Akiyo" video sequence and high-motion "Foreman" video sequence. Following this, Figure 6.10 shows the visual results of Shi-Wang-Bhargava Algorithm III (MVEA) for the same sequences.

Figure 6.9. Images (a) and (c) show the original MPEG frames from "Akiyo" and "Foreman", respectively. Images (b) and (d) show the visual defects caused by encrypting the sign bits of motion vectors in low-motion sequence "Akiyo" and high-motion sequence "Foreman", respectively.



(a) (b) (c) (d)

Figure 6.10. The leftmost two images show low-motion sequence "Akiyo" encrypted with Shi-Wang-Bhargava Algorithm III (MVEA) and decoded with: (a) the correct key, and (b) the incorrect key. The rightmost two images show high-motion sequence "Foreman" encrypted with Shi-Wang-Bhargava Algorithm III (MVEA) and decoded with: (c) the correct key, and (d) the incorrect key.

Table 6.5. Classification of Shi-Wang-Bhargava Algorithm III

Domain	MPEG videos
Perception-level	Medium
Cryptosystem	Fixed key (Vigenére-like)
Encryption Approach	Selective
Format Compliant	Yes
Bitrate	Near-constant

6.6.3 Shi-Wang-Bhargava Algorithm IV (RVEA)

Finally, the Algorithm IV (RVEA) is significantly more secure approach than the previous three algorithms. This approach is considered to be robust under both ciphertext-only attack and known-plaintext attack. The difference between RVEA and MVEA/VEA algorithms is that RVEA uses conventional symmetric key cryptography to encrypt the sign bits of DCT coefficients and the sign bits of motion vectors. The conventional cryptosystems are well mathematically understood, and thoroughly tested by the experts in the field, which definitely adds on to the security aspect of RVEA. The selective approach significantly speeds up the process of conventional encryption by only encrypting certain sign bits in the MPEG stream. The experiments show that the encryption quality is quite good considering the amount of information changed.

In the Algorithm IV (RVEA), the sign bits of DCT coefficients and motion vectors are simply extracted from the MPEG video sequence,

encrypted using a fast conventional cryptosystem such as AES, and then restored back to their original position in the encrypted form. The effect of this is similar to VEA/MVEA where the sign bits are either flipped or left unchanged. The authors limited the number of bits for encryption to at most 64 for each MPEG stream macroblock, for the purposes of reducing and bounding the computation time. Next, we describe exactly how these sign bits are selected for encryption.

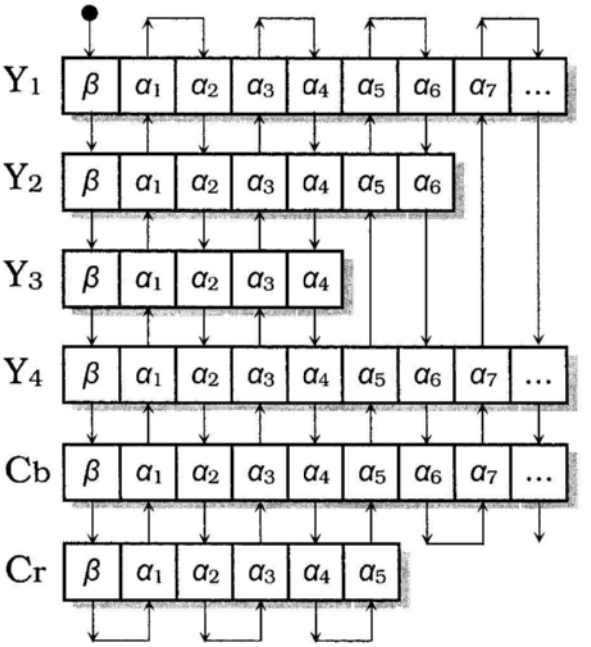
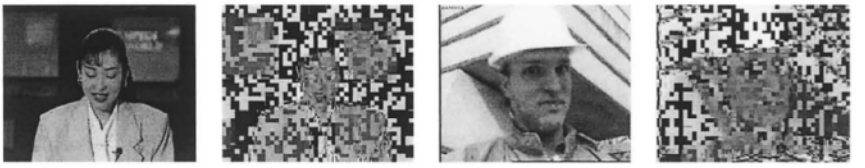


Figure 6.11. The bit selection order in RVEA (© 1999 IEEE).

Each MPEG video slice consists of one or more *macroblocks*. The macroblock corresponds to a 16x16 pixel square, which is composed of four 8x8 pixel luminance blocks denoted by Y_1, Y_2, Y_3 and Y_4, and two 8x8 chrominance blocks Cb and Cr. Each of these six 8x8 blocks can be expressed as a sequence $ba_1a_2...a_n$, where n is at most 64 since b is the code for DC coefficient and a_i is the code for a non-zero AC coefficient. Then, for each such 8x8 block, we can define the sequence $\beta\alpha_1\alpha_2...\alpha_n$, that corresponds to the sign bits of the matching DCT coefficients b, a_1, a_2, ..., and a_n. Figure 6.11 shows the order of selection of the sign bits for each macroblock. The

obvious reason for this selection order is that the DC and higher order AC coefficients are carrying much more information that the low-order AC coefficients.

In conclusion, RVEA only encrypts a fraction (typically about 10%) of the whole MPEG video by using conventional secure cryptographic schemes such as DES, IDEA, AES, etc. As such, it is more secure. Furthermore, it saves up to 90% of the computation time compared to the naïve approach. Like its predecessors, it is fully format compliant. Figure 6.12 shows the effects of RVEA.



(a) (b) (c) (d)

Figure 6.12. The leftmost two images show low-motion sequence "Akiyo" encrypted with Shi-Wang-Bhargava Algorithm IV (RVEA) and decoded with: (a) the correct key, and (b) the incorrect key. The rightmost two images show high-motion sequence "Foreman" encrypted with Shi-Wang-Bhargava Algorithm IV (RVEA) and decoded with: (c) the correct key, and (d) the incorrect key.

Table 6.6. Classification of Shi-Wang-Bhargava Algorithm IV.

Domain	MPEG videos
Perception-level	Medium
Cryptosystem	Conventional
Encryption Approach	Selective
Format Compliant	Yes
Bitrate	Near-constant

6.7 VIDEO ENCRYPTION BY ALATTAR, AL-REGIB AND AL-SEMARI

In 1999, Alattar, Al-Regib and Al-Semari presented three methods for selective video encryption based on DES cryptosystem [71]. These methods, called simply *Method I*, *Method II* and *Method III*, were

computationally improved versions of the previous work from two of the co-authors [70], which is referred to as *Method 0*.

The first algorithm (Method 0), proposed by Alattar and Al-Regib in [70], essentially encrypts all macroblocks from I-frames and the headers of all prediction macroblocks using DES cryptosystem. This method performs relatively poorly because encryption is carried out on 40%-79% of the MPEG video stream [62].

In Method I, the data of every n^{th} macroblock from the I-frame of MPEG video stream is encrypted using DES cryptosystem, while the information from the all other I-frame macroblocks is left unencrypted. The value of n was not specified, and it can be chosen depending on the application needs. If the value of n is 2 then the encryption is performed on approximately a half of all I-frame macroblocks, but the security level is higher. On the other hand, if the value of n is higher, the computational savings are bigger, yet the security level is lower. An important observation is that even though the certain number of I-macroblocks is left unencrypted, they are not expected to reveal any information about the encrypted ones [71]. However, this method does not hide the motion information, and the authors suggest an improved algorithm, which will be discussed next.

To improve the security of Method I, Alattar, Al-Regib and Al-Semari suggested Method II, which additionally encrypts the headers of all predicted macroblocks using DES. Since DES is a block cipher that operates on 64-bit blocks, a 64-bit segment starting from the header of a predicted macroblock is processed in the beginning. This segment may include exactly the whole header (which is the rare case when header size is equal to 64 bits), a part of the header (when header size is larger than 64 bits), or the whole header along with the part of the macroblock data (when the header size is smaller than 64 bits). In the case when the encrypted segment contains a part of the header, an adversary would have serious problems with synchronization, which adds to the security regarding motion vectors [71]. The security is further increased if the encrypted segment also contains a part of the macroblock data. The computation performed when using Method II is clearly faster then that of Method 0, but slower than that of Method I.

Finally, Alattar, Al-Regib and Al-Semari proposed Method III to reduce the amount of computation done in Method II. Namely, instead of encrypting all predicted macroblocks, the encryption in Method III is performed on every n^{th} predicted macroblock, along with encrypting every n^{th} I-macroblock.

Table 6.7. Classification of Alattar–Al-Regib–Al-Semari Algorithm.

Domain	MPEG videos
Perception-level	Variable
Cryptosystem	Conventional
Encryption Approach	Selective
Format Compliant	No
Bitrate	Near-constant

6.8 VIDEO ENCRYPTION BY CHENG AND LI

The partial encryption schemes for still images introduced by Cheng and Li are also further extended to the videos [40]. The approaches proposed by Cheng and Li are not suitable for JPEG image compression, and are thus naturally also not suitable for the MPEG video compression standard. Instead, the partial encryption algorithms are designed for video compression methods, which use either quadtree compression or wavelet compression based on zerotrees for the video sequence intraframes, motion compensation, and residual error coding. For example, partial encryption is applicable to the videos that are based on the Set Partitioning In Hierarchical Trees (SPIHT) image compression algorithm, which is an application of zerotree wavelet compression.

Cheng and Li's partial encryption algorithms are designed to disguise the intraframes (I-frames), the motion vectors, and the residual error code of the given video sequences. In both quadtree compression and wavelet compression based videos, all I-frames are encrypted using previously discussed methods for partial encryption of still images by Cheng and Li [40].

In addition, it is also important to encrypt motion vectors. If the motion vector information is unencrypted, the adversary may be able to use an image frame to obtain approximations to the successive

frames. Almost all motion estimation algorithms divide the frame into blocks and try to predict their movement (the position in the next frame) by constructing an estimated motion vector for each block. The blocks that belong to the same large object often have identical motion vectors and it is efficient to encode these vectors together. The authors restrict to those video encryption algorithms that use a quadtree to merge these blocks [40]. Then, quadtree partial encryption is used to encrypt the motion vectors.

Finally, for the security purposes it is important to encrypt the residual error as well. Unencrypted residual error may reveal the outline of a moving object. The residual error is often treated as an image frame and then compressed using some standard image compression algorithm. Again, we restrict ourselves to video compression algorithms that use either quadtree or wavelet based image compression algorithm to compress residual error frames. Thus, partial encryption schemes for both quadtree and wavelet compression can be applied to the residual error encryption.

Table 6.8. Classification of Cheng-Li Video Encryption Algorithm.

Domain	Quadtree compression and wavelet compression based videos
Perception-level	Low
Cryptosystem	Conventional
Encryption Approach	Selective
Format Compliant	No
Bitrate	Near-constant

6.9 VIDEO ENCRYPTION BY WU AND KUO

In their recent work [72,73], Wu and Kuo proposed two selective encryption algorithms for video sequences that use Huffman coder or QM coder: *MHT-encryption scheme* (where MHT stands for Multiple Huffman Tables), and *MSI-coder* (where MSI stands for Multiple State Indices).

6.9.1 Wu-Kuo Algorithm I

The MHT-encryption scheme is based on changing the standard Huffman table into a different one based on a secret key. Shi, Wang and Bhargava's Algorithm I, which was discussed above, proposed a similar idea. The basic procedure for generating the entropy code based on different Huffman tree (the "encryption") is summarized in the following three steps:

1. Generate 2^k different Huffman tables, numbered from 0 to $2^k - 1$.
2. Generate a random vector $P = (p_1, p_2, \ldots, p_n)$, where each p_i is a k-bit integer ranging from 0 to $2^k - 1$.
3. For the i^{th} symbol in the Huffman entropy encoding stage, use the Huffman table indexed by $p_{(i-1 \pmod n)+1}$ to encode it.

For the purpose of not affecting the compression ratio, it is important to select optimal Huffman tables that perform similarly to the standard one. This, of course, creates additional constraint that may be a security risk [39]. The authors propose creating a number of so-called training multimedia (images or audio data) that can be used to represent majority of the multimedia data in the world, so that the associated Huffman code that can be easily generated is optimal for the general use. The authors' experimental results show relatively small increase in the output images on average, which means that in this approach the compression performance will be satisfactory, even though the symbols are chosen from the multiple Huffman tables.

Figure 6.13. The Huffman tree mutation process.

Another method for obtaining optimal Huffman trees is proposed in [72,73]. Namely, only 4 base optimal Huffman trees are generated, while the other optimal tables are produced by what is referred to as a *Huffman tree mutation process*. The label pairs are permuted using

the secret permutation, as illustrated by the example in Figure 6.13. Clearly, there is no change in the optimality. In addition, the authors propose two slight security enhancements for the MHT-encryption scheme to increase the resistance against known-plaintext and chosen-plaintext attacks [73].

Table 6.9. Classification of Wu-Kuo Video Encryption Algorithm I.

Domain	Video codecs based on Huffman coder
Perception-level	Low
Cryptosystem	Huffman tree replacement based on a secret key
Encryption Approach	Selective
Format Compliant	No
Bitrate	Increased

6.9.2 Wu-Kuo Algorithm II

The second method (MSI-coder) is designed for video codecs that use QM coding instead of the Huffman coding. The QM coder is an adaptive coder based on low-cost probability estimation. It dynamically adjusts the statistical model to a received binary sequence symbols, whose probability is updated at a low computational cost via table look-up. In the standard QM coder there are a total of 113 possible values for the state index, whose initial value is 0 indicating the equal probabilities of 0's and 1's. Since there are only 113 possibilities to initialize the state index, initializing it to a different "secret" value is useless. Therefore, the MSI-coder uses four indices, which are initially set to different secret values, and used alternatively in a secret order. MSI-coder consists of the following four steps:

1. Generate a random key K = {(s_0, s_1, s_2, s_3), (p_0, p_1, ..., p_{n-1}), (o_0, o_1, ..., o_{n-1})}, where each s_i is a 4-bit integer, while each p_i and o_i is a 2-bit integer.
2. Initialize four state indices (I_0, I_1, I_2, I_3) to (s_0, s_1, s_2, s_3).
3. To encode the i^{th} bit from the input, we use the index $Ip_{i \pmod n}$ to determine the probability estimation value Q_e.
4. If the state update is required after encoding the i^{th} bit from the input, all state indices are updated except for $Io_{i \pmod n}$.

The not-so-obvious step 4 is included to ensure that the state indices will become different even if they become the same at some point. By experiments, Wu and Kuo showed that the MSI encoded data is increased in size by only 0.82% - 4.16% when compared to the data encoded with the original QM coder. Like the previous scheme, the MSI-coder can similarly be slightly enhanced against the known-plaintext and chosen-plaintext attacks [73].

Table 6.10. Classification of Wu-Kuo Video Encryption Algorithm II.

Domain	Video codecs based on QM coder
Perception-level	Low
Cryptosystem	Conventional pseudo-random number generator
Encryption Approach	Selective
Format Compliant	No
Bitrate	Near-constant (up to 5% increase)

6.10 VIDEO ENCRYPTION BY ZENG AND LEI

Zeng and Lei proposed selective encryption algorithms for digital videos based on wavelet framework and the 8×8 DCT framework [74]. In their schemes, the transform coefficients are divided into blocks or segments that, together with the motion vectors, undergo selective encryption.

6.10.1 Zeng-Lei Algorithm I

Zheng and Lei stressed out the importance of preserving format-compliance as well as preserving the compression ratio [74]. In the case of the wavelet-based systems, the input video frames are transformed using the wavelet filter banks. By using separable filtering along both the horizontal and vertical directions, we obtain wavelet decomposition and wavelet subbands of a frame. Figure 6.14-b shows 5-level wavelet decomposition of "Lena" that produces 16 subbands. Since the coefficients of the subbands are arranged in the spatial arrangement of the original image, high correlation exists between neighboring coefficients. In addition, since each subband represents selected spatial frequency information of the input frame, the distribution of the coefficients differs at the inter-subband level. Zheng-Lei Algorithm I uses two methods to scramble and shuffle

these coefficients, so that these statistical properties are for the most part not affected. This way the encryption algorithm preserves the compression ratio.

(a) (b)

Figure 6.14. Wavelet decomposition of a frame "Lena": (a) the original frame, (b) 5-level wavelet decomposition of the frame that produces 16 subbands (© 2002 IEEE).

In the first method, Zheng and Lei chose to encrypt only the bits of individual transform coefficients with high entropy. Since the coefficients that posses high entropy are consequently also highly unpredictable, encrypting them will not significantly affect the compression performance. In general, Zheng and Lei observed that each coefficient could be separated into three groups of bits: significance bits, refinement bits, and a sign bit. *Significance bits* are the most significant bit with a value of 1 together with any preceding bits with a value of 0. The significance bits are the worst candidates for selective scrambling since they have the lowest entropy, and therefore are highly compressible. The *sign bit* simply specifies whether the magnitude of a coefficient is positive or negative. Sign bits are not highly compressible, and as such, they represent the best candidates for selective encryption. Finally, the *refinement bits* represent the rest of the magnitude bits (the bits that do not belong to the group of significance bits). The refinement bits are somewhere in the middle as far as the compressibility, and thus they can be scrambled too, but the degradation level is not as high as in the sign bit scrambling. Zheng-Lei Algorithm I mandatory

encrypts the sign bits, but optionally encrypts the refinement bits of the coefficients. When encrypting refinement bits, to reduce complexity, only the most significant or the two most significant refinement bits undergo the encryption process.

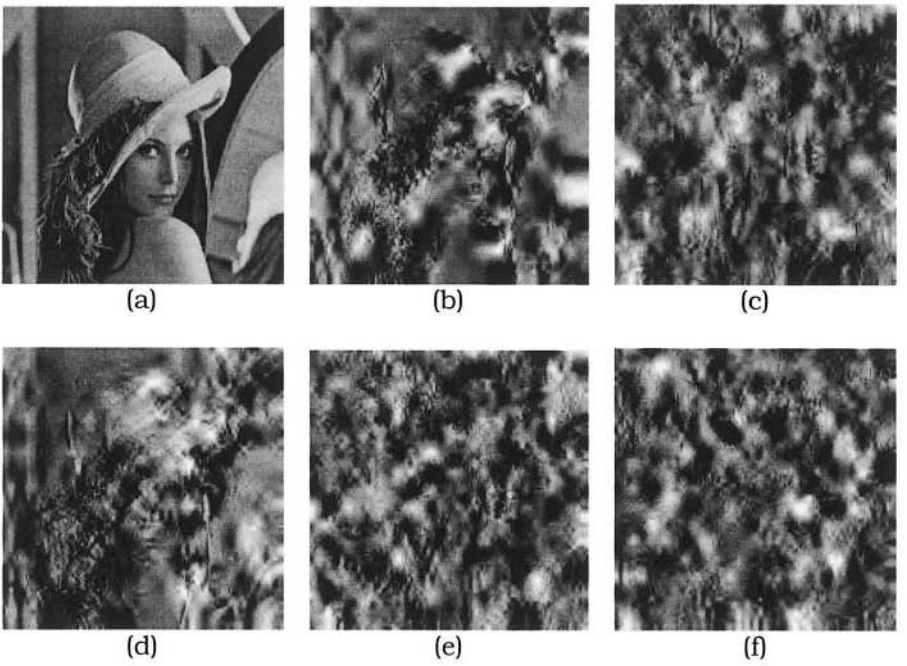


Figure 6.15. Encryptions of "Lena" using the transformations of Zeng-Lei Algorithm I: (a) the original frame, (b) encrypted sign bits only, (c) permuted blocks, (d) rotated blocks, (e) encrypted sign bits and permuted blocks, and (f) encrypted sign bits, permuted blocks, and rotated blocks (© 2002 IEEE).

The second method proposed by Zeng and Lei permutes blocks of coefficients. Each subband is divided into a number of blocks of equal size; however, the size of a block can vary for different subbands. The algorithm uses tables of permutations that can differ on both the inter-subband and inter-frame levels. A particular shuffling table is generated using a secret key. Shuffling of the blocks preserves intra-block statistics, so that the algorithm retains the compression ratio. Furthermore, shuffling of the blocks of coefficients changes the high-level spatial configuration of the video

frequency content, which is much harder to cryptanalyze than the local intra-block shuffling that was utilized in Tang Algorithm [33]. Block sizes are application dependent, however, using larger and fewer blocks results in less degradation.

An additional operation that could be utilized on the block level is rotation. Recall that there are total of eight possible rotation operations that could be utilized on a two dimensional block. Each such rotation preserves the statistical properties within a block, thus allowing for the unaffected compression efficiency. In Zeng-Lei Algorithm I, the rotation is selected according to the secret key.

From the theoretical point of view, the algorithm is quite effective, in that the brute-force attacks are computationally infeasible. However, the Zeng-Lei Algorithm I is still quite vulnerable to the known/chosen-plaintext attacks [7].

Table 6.11. Classification of Zeng-Lei Algorithm I.

Domain	Wavelet compression based videos
Perception-level	Medium
Cryptosystem	Conventional
Encryption Approach	Selective
Format Compliant	Yes
Bitrate	Near-constant

6.10.2 Zeng-Lei Algorithm II

The second algorithm by Zheng and Lei selectively encrypts the DCT-based compression video standards such as MPEG or H.26X. In all such standards the video sequence is segmented into a group of pictures (GOP), where each GOP has an intra-coded frame (I-frame) followed some forward predictive coded frames (P-frames) and bi-directional predictive coded frames (B-frames). The lowest layer represents an 8×8 block that is a unit of DCT transform coding. The 2×2 square of luminance blocks and the corresponding chrominance blocks together form a *macroblock*, while a horizontal strip of adjacent macroblocks forms a *slice*. Each layer except for the block layer has an associated header segment within the bitstream. According to the Zeng-Lei Algorithm II, the I-frames are first divided into segments. Each of the segments is consisted of several

macroblocks or blocks, chosen sequentially or randomly. Within each segment, DCTs of the same frequency location are treated as being part of a "subband". As such, the coefficients within the same subband can be scrambled in the same way as in Zeng-Lei Algorithm I. Within each slice, the DC coefficients are permuted according to a table of permutations and a secret key, while the AC coefficients of the same frequency location are permuted using different shuffling tables.

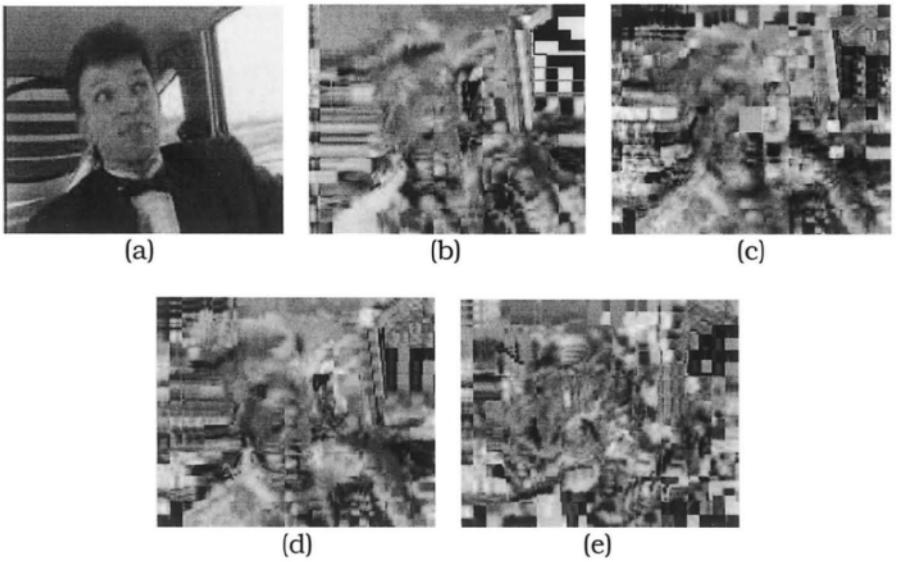


Figure 6.16. Encryptions of "Carphone" video sequence using the transformations of Zeng-Lei Algorithm I: (a) the original frame, (b) encryption of sign bits in I- and P-frames, (c) encryption of sign bits and slice permutation in I- and P-frames, (d) encryption of sign bits and slice permutation in I-frames and encryption of sign bits in P-frames, and (e) encryption of sign bits and slice permutation in I-frames and encryption of sign bits and motion vectors in P-frames.

This procedure could be applied to the chrominance coefficients alike. In addition, the algorithm encrypts the sign bits of the coefficients. For non-intra-coded blocks, we encrypt the sign bits of DC and AC coefficients, which results in a pseudo-random sign bit value flip. The DC coefficients of intra-coded blocks always have positive value, and for those coefficients, the algorithm pseudo-randomly flips their value with respect to a threshold. Furthermore,

the Zeng-Lei Algorithm II requires shuffling of DCT coefficients of the intra-coded blocks of P- and B-frames, and recommends shuffling of DCT coefficients of the non-intra-coded blocks of P- and B-frames.

The algorithm also requires encrypting the motion vector information to further degrade the video. Even when all of the I-frames and all of the intra-coded blocks are encrypted, the motion patterns are still clearly visible if the motion vectors are left unencrypted. Thus, the sign bits of motion vectors should be encrypted in the same manner the sign bits of DCT coefficients are. In addition, the motion vectors should be shuffled within a segment. Figure 6.16 shows the scrambling effects of the algorithm. Zeng-Lei Algorithm II is secure against ciphertext-only attack, however, it is not particularly secure against known/chosen-plaintext attacks. In addition, the algorithm is fully format compliant and has a near-constant bitrate.

Table 6.12. Classification of Zeng-Lei Algorithm II.

Domain	DCT compressed videos (MPEG, H.26X)
Perception-level	Medium
Cryptosystem	Conventional
Encryption Approach	Selective
Format Compliant	Yes
Bitrate	Near-constant

SUMMARY

Most video encryption methods are focused on selectively encrypting only certain bits from the compressed domain. For example, many approaches try to disguise some information about transform coefficients or motion vectors. Unfortunately, even though the conventional cryptosystems are used as a basis of many of the proposed algorithms, a high security is not always guaranteed since the unencrypted data often strongly correlates to the encrypted data thus opening doors for the successful cryptanalysis.

Chapter 7

SPEECH AND AUDIO ENCRYPTION

In many applications, the audio sequences need confidentiality during transmission. Sometimes it is enough to apply the naïve approach, but in many instances, this is too computationally expensive (for example, in small mobile devices). As far as security is concerned, perhaps the most important type of audio data is speech. Unlike in the case of music files and similar entertainment audio sequences, in many applications speech requires substantial level of security. This chapter discusses the methods for speech protection, as well as the protection of general audio compressed sequences such as MP3.

7.1 SPEECH AND AUDIO SCRAMBLING

Traditional speech scrambling has a long history. As in the case of analog video signals, the early work in this area was based on analog devices that simply permute speech segments in the time domain or distort the signal in the frequency domain by applying inverters and filter banks. These schemes are very insecure by today's computing standards. Today's research has shifted towards securing speech in the digital form, by applying selective encryption.

Furthermore, there are numerous applications where the general audio data needs to be protected, and the expensive naïve approach might be infeasible. This demand created some research initiatives towards developing selective encryption approaches for compressed audio streams.

7.2 SPEECH ENCRYPTION BY WU AND KUO

One of the most popular digital speech codecs is the ITU recommendation G.723.1 compression standard [75]. It has a very low bit rate, and it is extremely suitable for voice communications over packet-switching based networks. It is also a part of the ITU

H.324 standard for video-conferencing/telephony over regular public telephone lines.

The compression is based on the analysis-by-synthesis method. The encoder incorporates the entire decoder unit, which synthesizes a segment of speech according to given input coefficients. The encoder then changes these coefficients until the difference between the original speech and the synthesized speech is within the acceptable range. The decoding is performed with three different decoders: LSP decoder, pitch decoder and excitation decoder, and the G.723.1 coefficients can be categorized depending on the decoder they are fed in to. In addition, this codec can work in either 6.3 Kbps or 5.3 Kbps modes.

In [72], Wu and Kuo suggest applying selective encryption to the most significant bits of all important G.723.1 coefficients. They identified the following coefficients as the important ones: the LSP codebook indices, the lag of pitch predictor, the pitch gain vectors, the fixed codebook gains, and the VAD mode flag. The total number of selected bits for encryption is 37 in each frame, which is less than 1/5 of the entire speech stream at the 6.3 Kbps rate, and less than 1/4 of the entire speech stream at the 5.3 Kbps rate.

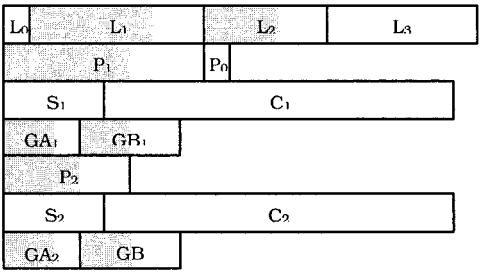
Table 7.1. Classification of Wu-Kuo Speech Encryption Algorithm.

Domain	G.723.1 speech
Perception-level	Medium
Cryptosystem	Conventional
Encryption Approach	Selective
Format Compliant	No
Bitrate	Near-constant

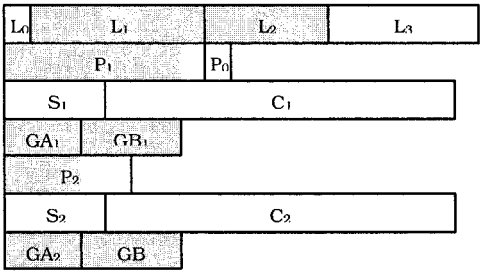
7.3 SPEECH ENCRYPTION BY SERVETTI AND DE MARTIN

In 2002, Servetti and De Martin published a perception-based algorithm for partial encryption of telephone bandwidth speech [76]. The algorithm was implemented for the ITU-T G.729 codec for a rate of 8 Kbps [77].

Servetti and De Martin suggested two methods for partial telephony speech encryption. The first method was intended to have low-security, but high bit rate (similar to the degradation schemes for videos and images). The goal of this method was to degrade the speech enough to prevent immediate eavesdropping. However, more substantial cryptanalysis could reveal the encrypted speech. The second algorithm encrypts more of the bit-stream, and its goal is to provide more security. Servetti and De Martin argue that even though it encrypts only about a half of the bitstream, the algorithm's security level is comparable to that of the naive algorithm.



(a)



(b)

Figure 7.1. Selective encryption for G.729 speech codec (grey bits are selected for encryption): (1) low security algorithm, and (2) high security algorithm.

There are total of 15 segments from the ITU-T G.729 codec output: L_0 (1 bit), L_1 (7 bits), L_2 (5 bits), L_3 (5 bits), P_1 (8 bits), P_0 (1 bit), S_1 (4 bits), C_1 (13 bits), GA_1 (3 bits), GB_1 (4 bits), P_2 (5 bits), S_2 (4 bits), C_2 (13 bits), GA_2 (3 bits), and GB_2 (4 bits). Figure 7.1 depicts the selected bits for encryption in Servetti and De Martin's low and high

security algorithms. The experiments showed that by applying the high security selective encryption algorithm, the speech is highly scrambled, it cannot be reconstructed from the known bits, and the total computational saving is more than 50%.

Table 7.2. Classification of Servetti-De Martin Speech Encryption Algorithm.

Domain	G.729 speech
Perception-level	Variable
Cryptosystem	Conventional
Encryption Approach	Selective
Format Compliant	No
Bitrate	Near-constant

7.4 AUDIO ENCRYPTION BY THORWIRTH, HORVATIC, WEIS, AND ZHAO

Thorwirth, Horvatic, Weis, and Zhao proposed a selective encryption algorithm for perceptual audio coding (PAC) based compression standards, such as MPEG-1 Layer III (MP3) [78]. In their proposal, the authors concentrated on analyzing encryption of MP3 encoded audio files.

In the late 1980's, the researchers at Fraunhofer Institute developed the audio coding technique called MPEG-1 Layer III, or simply MP3. This codec achieved the stunning compression ratio of 1:10 - 1:12, while still maintaining the CD sound quality. It is a perception audio codec (PAC) that uses the perceptual masking algorithms to discard the redundancy from the raw audio digital signal. All components of the raw audio signal that are inaudible to the human hearing system are identified as redundant data by the perceptual models. The compression in MP3 audio standard is based on the Huffman entropy code. More information on MP3 standard could be found in [80,81].

Hearable frequency spectrum is divided into the segments called *audio quality layers*. The 20Hz - 4kHz range represents the lowest audio quality layer, while the CD quality range is 20Hz-22kHz. The authors propose encrypting the audio quality layers associated with

the compressed audio data, in order to enforce audio quality control by means of encryption. The algorithm identifies the frequency spectrum boundaries that are used to determine the audio quality layers. Each known layer is then grouped into the blocks of equal size and a block cipher is applied. After the encryption stage, the encrypted data can be easily plugged back to the MP3 bit-stream, thus preserving the format. Such encrypted MP3 sequences are decodable and playable by any valid MP3 player [78].

The approach by Thorwirth, Horvatic, Weis, and Zhao is format-compliant, however, due to the block cipher usage, there may be a slight increase in the bitstream (near-constant bitrate).

Table 7.3. Classification of Thorwirth-Horvatic-Weis-Zhao Audio Encryption Algorithm.

Domain	MP3 audio
Perception-level	High
Cryptosystem	Conventional
Encryption Approach	Selective
Format Compliant	Yes
Bitrate	Near-constant

7.5 AUDIO ENCRYPTION BY SERVETTI, TESTA AND DE MARTIN

In 2003, another selective encryption of MP3 audio files was proposed. Servetti, Testa, and De Martin proposed a format-compliant audio degradation method [79] that distorts the quality of sound of MP3 sequences, but preserves the perceptual information. The effect of the algorithm is a *sample-mode* music that could be upgraded to a high-quality music only by applying a proper decryption.

During MP3 compression each output channel is subdivided into 18 bands using a windowed *modified discrete cosine transform* (MDCT). The MDCT coefficients are encoded using a Huffman table and then partitioned into five regions: region0, region1, region2, count1, rzero (see Figure 7.2-a)

These regions are part of a logical structure called *Huffman codebits*. The highest independent logical component of an MP3 bitstream is an *MP3 frame*. A frame consists of a *header*, *audio data* field, *main data* field, and *ancillary data*. The header contains the decoding parameters such as the bitrate, the sampling frequency, and the number of channels, and also the synchronization bits, system information and an optional 16-bit cyclic redundancy check (CRC) code. The audio data field specifies the decoding control parameters and the side information about bit allocation and scalefactors. The main data field is located at the end of the audio data field and it contains scalefactors followed by the Huffman codebits. The end of each frame contains a segment of ancillary data. Figure 7.2-b shows the structure of an MP3 frame.

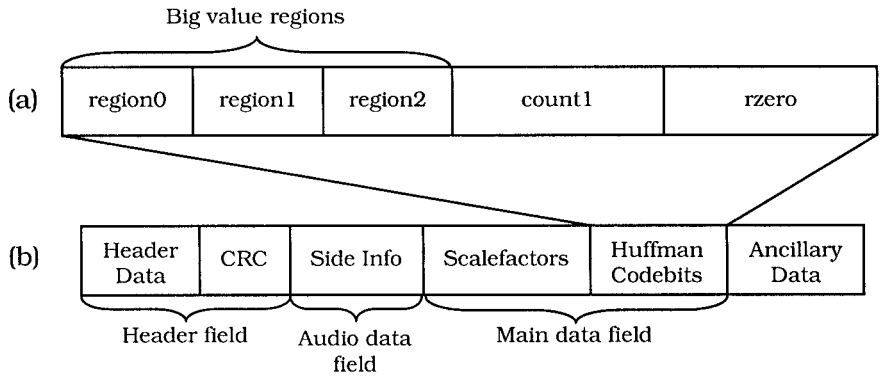


Figure 7.2. The MPEG-1 Layer III (MP3) bitstream structure: (a) the five regions of Huffman codebits, and (b) the MP3 frame.

Because of the psychoacoustic quantization model that is applied to the coefficients, the large values of MDCT coefficients occur at the low spectral frequencies, while low values and zeros occur at the high spectral frequencies. A total of 576 coefficients are then partitioned into the aforementioned five regions according to the following rules. Starting from the highest frequencies, all adjacent pairs of zero values are partitioned into the rzero region. Next, all adjacent quadruples having values 0, 1, or -1 are assigned to the count1 region. Finally, the remaining pairs whose values are in the range [-8191, 8191] are assigned to the last three regions that are also referred to as the *big value regions*. From a pool of 32 Huffman tables (out of which only 30 are used), a different Huffman table is

associated with each of the three big value regions to further enhance the compression and error resilience. The count1 region is offered a choice of two different Huffman tables, while the rzero region has no associated Huffman encoding.

Most MP3 encoders map an audio range from 20Hz to 14kHz into the big value regions. In most cases, for a 44.1kHz sample rate, 0-2kHz signal is mapped to a region0, 2-5kHz signal is mapped to a region1, and 5-14kHz signal is mapped to a region2. Then, the Huffman table is chosen that best fits the statistics of a given region. The Huffman table index from 0 to 31 is stored for each region in the so-called *table select* value of the corresponding audio data field.

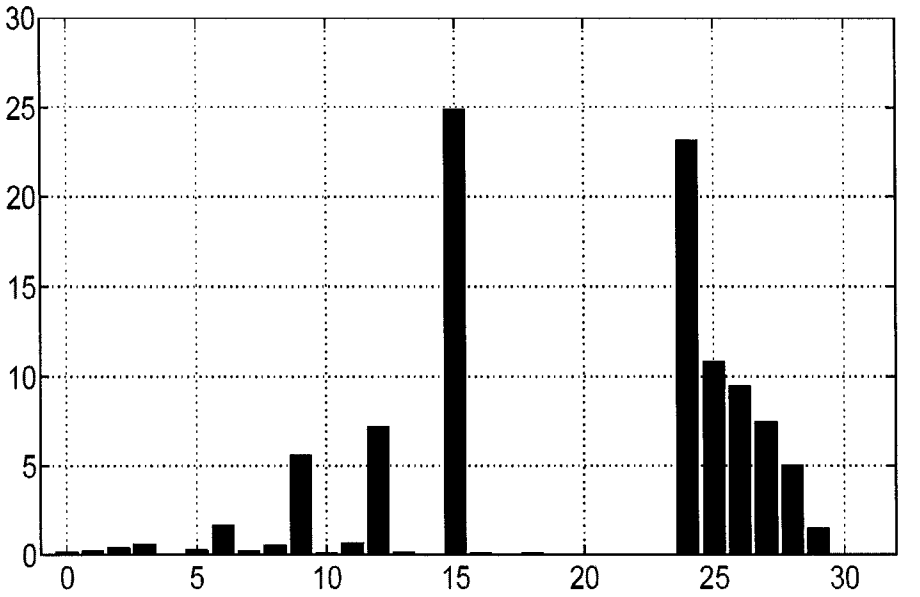


Figure 7.3. Average usage of the Huffman tables for big value regions (© 2003 IEEE).

Servetti, Testa, and De Martin noticed that it is possible to instruct a standard MP3 decoder to ignore the spectral values of a given frequency region using a *private Huffman table index*. Namely, the MP3 standard reserves the usage of possibly two additional Huffman tables for encoding the big value regions. Either 4 or 14 could index these two tables, while the rest of the indices in the range 0 to 31 are used to index the standard 30 Huffman tables for big value regions.

Although there is no strictly defined behaviour of a decoder when a reserved index is encountered, most decoders fill the corresponding portion with zero value coefficients. In the MP3 selective encryption algorithm proposed by Servetti, Testa, and De Martin, the Huffman table indices of all regions after region0 are set to the one of the unused values, either 4 or 14. This way, the decoder will skip the decoding process of all four regions that follow region0.

However, to preserve the format compliance afterwards, an additional modification has to occur as well. Namely, in order for the decoder to safely discard the remaining bits in the main data field, all of the number of big value pairs must be set to the maximum value of 288. Such bitstream is properly decidable and playable by the standard MP3 decoder, however the sound is degraded to a lower quality when only region0 is decoded. The enhancement information, that includes the correct table indices as well as the correct number of big value pairs for all given MP3 frames, are encrypted using a conventional cryptosystem and attached at the beginning of the MP3 bitstream. Even though this prefix is not a part of the MP3 file format, most standard MP3 players will simply ignore it and process the rest of the bitstream. The increase in the bitstream caused by the encrypted information is very minimal, and in most instances accounts for less than 5%. Unfortunately, the frequency distribution of the average Huffman tables used for big value regions is not uniform at all. Figure 7.3 shows that tables 15 and 24 are used more than a half of the time.

Consequently, a cryptanalytic attack could be mounted where an attacker replaces all of the table select values with 15 to obtain a sound of possibly much better quality, since the number of big value pairs is also prone to statistical cryptanalysis [79]. Thus, additional bits need to be encrypted. The authors suggested encrypting the part of data that an attacker would try to decode if the Huffman indices and big value numbers were successfully recovered by the attacker. Therefore, we must also encrypt one bit out of twenty in the region1 (about 1.1% of the share), and 70-100 bits in region2 (about 6.3%-8.3% of the share). The scheme is for the most part format-compliant, and the amount of encrypted information is only up to 10%. This is a low-security encryption (degradation), but in most instances, MP3 data usually calls for quality control encryption as opposed to absolute security.

Table 7.4. Classification of Servetti-Testa-De Martin Audio Encryption Algorithm.

Domain	MP3 audio
Perception-level	High
Cryptosystem	Conventional
Encryption Approach	Selective
Format Compliant	Yes
Bitrate	Near-constant

SUMMARY

There are applications where naïve encryption of speech and audio is not suitable. Most research efforts were concentrated on selectively encrypting the speech with a more secure approach, and selectively encrypting the audio and music using less secure, degradation approaches. For the MP3 standard, selective encryption offers an excellent quality control tool.

Chapter 8

VISUAL AND AUDIO SECRET SHARING

Secret sharing is an important concept in modern cryptography. Often, it is desired that only a certain group of people can recover the secret. This powerful concept can be extended to visual images and more recently, even to audio signals.

8.1 THE CONCEPT OF SECRET SHARING

The concept of secret sharing was independently introduced by George Blakeley [82] and Adi Shamir[5] [83] in 1979. Although conceptually the same, the two schemes are based on completely different mathematical structures. Shamir's secret sharing scheme is based on Lagrange polynomial interpolation, while Blakeley's scheme is based on vector spaces. We do not present the details of Shamir's and Blakeley's schemes here, however, an interested reader should examine [82] and [83] for the complete discussion.

A *secret sharing scheme* for a set P of n participants can be defined as a method for encoding a secret S into n shares, so that each participant from P receives exactly one unique share, and only certain qualified subsets of P can recover S from the information provided by their shares. A k *out of* n secret sharing scheme, where $2 \leq k \leq n$, is a secret sharing scheme in which the qualified subsets are all subsets of cardinality at least k. A trusted party that distributes the shares is called the *dealer*. Naturally, it is assumed that the shares are securely distributed by the dealer. A secret sharing scheme, which is sometimes also referred to as a *threshold scheme*, is called *perfect* if non-qualified subsets cannot gain any information whatsoever about the secret S by examining and analyzing their shares. A share is sometimes referred to as *shadow*, which is a term that Blakeley has used.

---

[5] Adi Shamir is also 'S' in RSA, and one of the co-inventors of differential cryptanalysis that is applicable to DES cryptosystem.

This concept was first applied to numbers, but in the 1990s, the researchers extended it to images and audio signals. Next, we discuss visual cryptography, which implements secret sharing where the shared secret S is an image.

8.2 VISUAL CRYPTOGRAPHY

The clever idea of *visual cryptography* to facilitate secret sharing of pictures is due to Naor and Shamir [84]. The pixels of secret picture are treated as individual secrets to be shared using a secret sharing (threshold) scheme. The picture is split into two or more shared images that on a transparent paper join into the original image. Clearly, this involves the same concept that was around since 1979.

Visual cryptography was introduced by Naor and Shamir at the EUROCRYPT '94 [84]. Naor and Shamir considered the general k out of n visual cryptography schemes, denoted by (k, n)-VCS, and gave optimal constructions for (2, n)-VCS and (n, n)-VCS. However, their constructions for cases when $2 < k < n$ were quite inefficient. These inefficient constructions were improved by Droste in [85] and by Ateniese, et al, in [86]. In addition, the constructions of visual schemes for general qualified subsets of [1, n] were proposed by Ateniese, Blundo, De Santis and Stinson [86]. To maintain a reasonable scope of this book, we only discuss Naor and Shamir's (2, n)-VCS and (n, n)-VCS, and Droste's (k, n)-VCS when $2 < k < n$. The beautiful thing about a visual secret sharing scheme is that it relies on a human visual system to perform the decryption. That is, the decryption does not involve any computation or computational device.

8.2.1 The Basic Idea: Constructing a (2, 2)-VCS

The basic idea of visual cryptography can be best described by considering a (2,2)-VCS case. Let us consider a binary image S containing exactly m pixels. The dealer creates two shares (binary images), S_1 and S_2, consisting of exactly m pixels, but each pixel further consists of two subpixels. Each pixel is encoded individually, and the encoding process is illustrated on Figure 8.1.

The dealer randomly determines one of the encoding schemes. Notice that it is equally likely that a shared pixel was originally black or white. Therefore, the scheme is perfect. Only by joining their

shares, the two participants can recover the pixel: if the superposition of the two shared subpixels results in two black subpixels (one black pixel), the original pixel was black; otherwise, the superposition creates one black and one white subpixel within a pixel, which indicates that the original pixel was white.

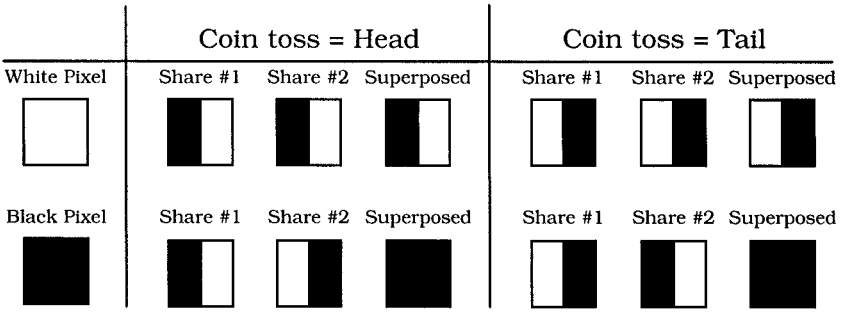


Figure 8.1. The encoding of a binary pixel in (2, 2)-VCS. The dealer randomly tosses a coin, which determines the encoding scheme.

Obviously, the picture is slightly modified since a white pixel becomes half-white, however, the human visual system can still recognize the image. This is referred to as the *loss of contrast.* In any visual secret sharing scheme, some loss of contrast will always occur. However, for more advanced schemes which have more subpixels involved the loss of contrast is more obvious. Figure 8.2 shows the sharing of "Lena" in (2, 2)-VCS.

Mathematically, we can represent the white pixel by 1 and the black pixel by 0. The (2, 2)-VCS from above can be represented in a Boolean matrix form. Let C_0 and C_1 be the following two collections of matrices:

$$C_0 = \left\{ \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \right\}, \text{ and } C_1 = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right\}.$$

To share a white pixel, the dealer randomly selects one of the matrices in C_0, and to share a black pixel, the dealer randomly selects one of the matrices in C_1. The first row of the chosen matrix is used for share S_1, and the second for share S_2. For example, a row (1 0) defines the black-white scheme in a share. Throughout

this chapter, C_0 and C_1 will denote the collections of matrices used to randomly select a share scheme to encode a white or black pixel (respectively) from the original image.

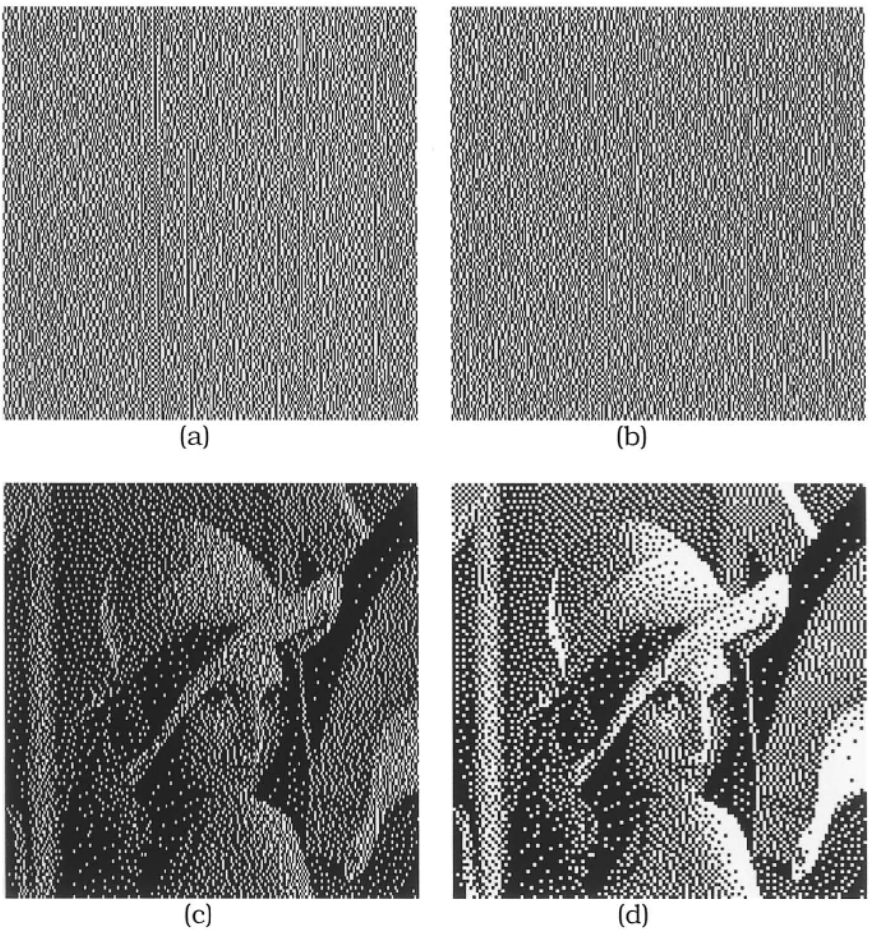


Figure 8.2. A (2, 2)-VCS of binary "Lena": (a) the first share S_1, (b) the second share S_2, (c) superimposed S_1 and S_2, and (d) the original image ("Lena" obtained by Floyd-Steinberg binary approximation).

In general, a solution to the k out of n visual cryptography scheme consists of two collections of $n \times l$ Boolean matrices C_0 and C_1. The collections C_0 and C_1 constitute a (k, n)-VCS if for some nonnegative integers h and m, $h > m$, the following conditions are met:

- The bitwise OR x of any k of the rows in C_0 satisfies $H(x) \leq l - h$.
- The bitwise OR x of any k of the rows in C_1 satisfies $H(x) \geq l - m$.
- For any $i_1 < i_2 < \ldots < i_s$ in $[1, n]$ with $s < k$, the matrices obtained by restricting C_0 and C_1 to rows $i_1, i_2, \ldots, i_s$ are equal up to a column permutation.

For the general k out of n sharing, finding such collections involves some challenging combinatorial problems. Fortunately, due to many research contributions over the last decade, we now have optimal and nearly optimal constructions of such collections.

8.2.2 Constructing a (2, *n*)-VCS

The general problem of constructing a 2 out of n visual cryptography scheme can be solved with the following two $n \times n$ matrix collections:

$$C_0 = \left\{ \pi\left(\begin{bmatrix} 0 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & 1 & \cdots & 1 \end{bmatrix} \right) \right\}, \text{ and } C_1 = \left\{ \pi\left(\begin{bmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ 1 & 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 0 \end{bmatrix} \right) \right\},$$

where $\pi(M)$ represents the collection if all matrices obtained by permuting the columns of a matrix M.

The dealer encodes the pixel b into n subpixels in each of the n shares. It is easy to observe that the bitwise OR of any two or more rows of a matrix from C_0 contains a white subpixel. On the other hand, the bitwise OR of any two or more rows of a matrix from C_1 does not contain a white subpixel. This is the key feature that distinguishes the white pixel from the black in the decoding process by the human visual system. Clearly, having only one share does not reveal any information whatsoever about the secret image, thus making the scheme *perfect*. The above construction of (2, n)-VCS was proposed by Naor and Shamir in [84].

The following figure shows a 2 out of 3 sharing of "Lena" obtained by using the collections C_0 and C_1 defined above. The dealer creates a total of three shares. By superimposing any two of these three shares reveals the image of "Lena". On the other hand, analyzing the shares individually does not reveal any information about "Lena" (a *perfect* secret sharing scheme).

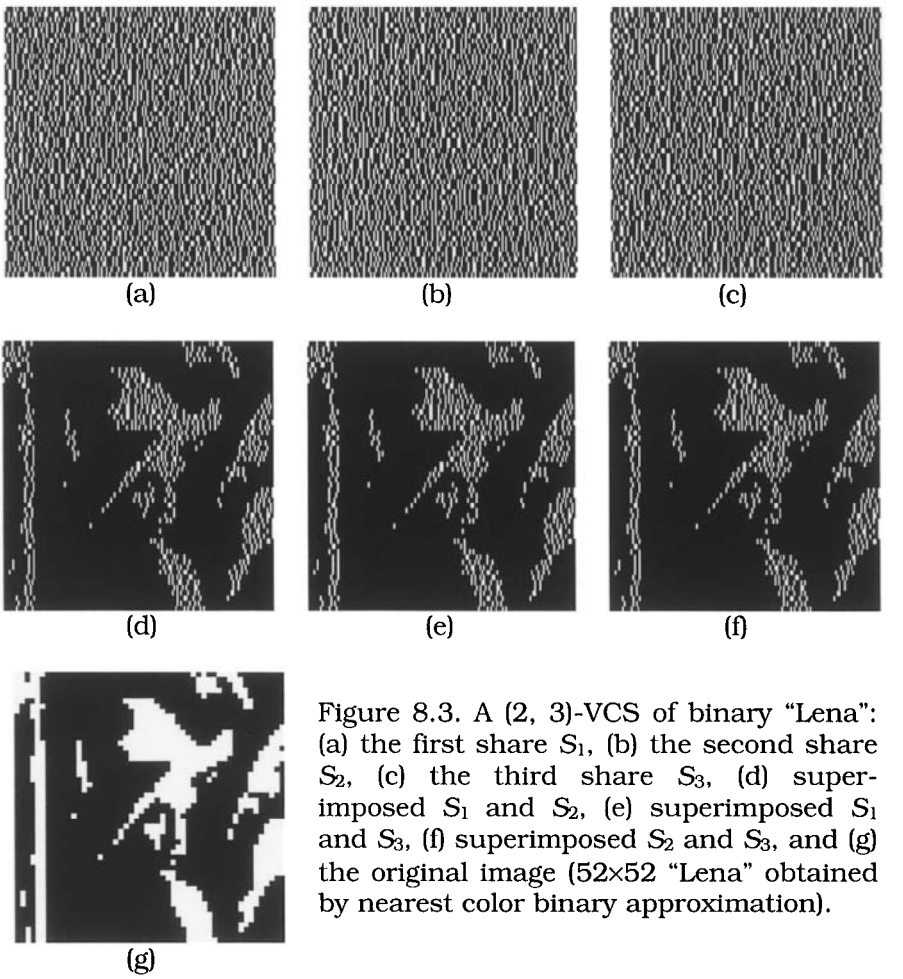


Figure 8.3. A (2, 3)-VCS of binary "Lena": (a) the first share S_1, (b) the second share S_2, (c) the third share S_3, (d) superimposed S_1 and S_2, (e) superimposed S_1 and S_3, (f) superimposed S_2 and S_3, and (g) the original image (52×52 "Lena" obtained by nearest color binary approximation).

8.2.3 Constructing an (*n*, *n*)-VCS

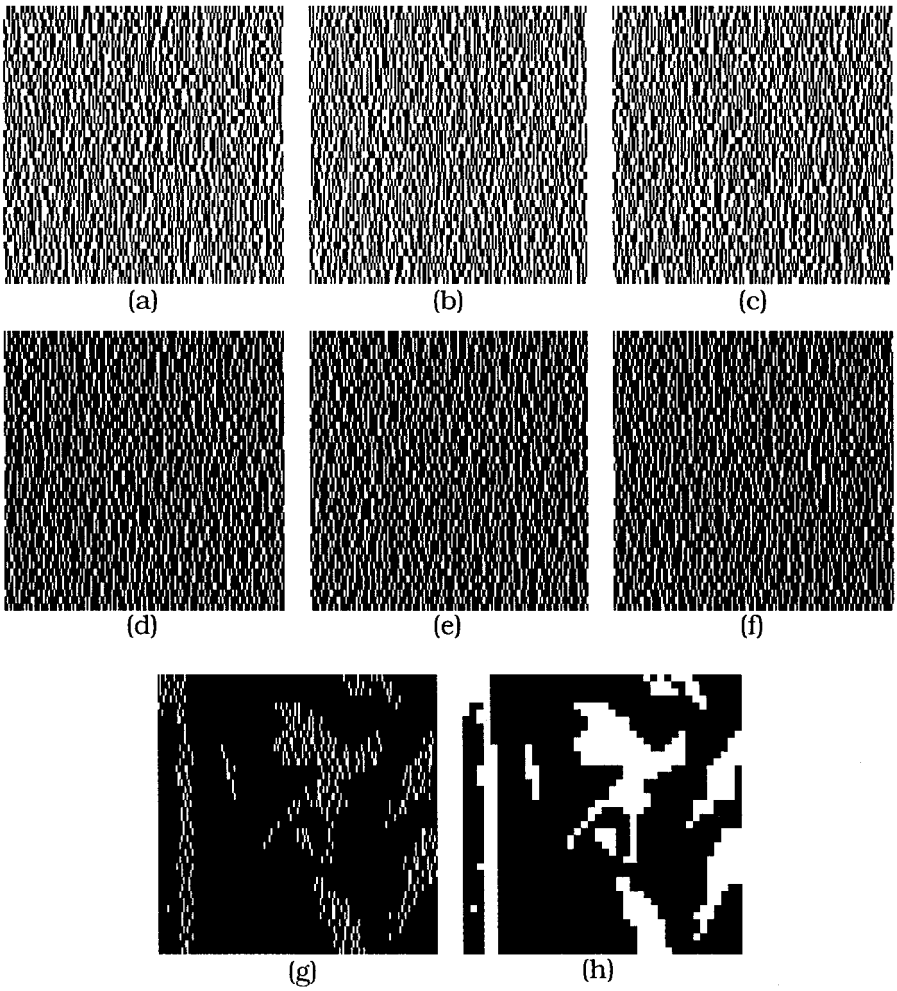


Figure 8.4. A (3, 3)-VCS of binary "Lena": (a) the first share S_1, (b) the second share S_2, (c) the third share S_3, (d) superimposed S_1 and S_2, (e) superimposed S_1 and S_3, (f) superimposed S_2 and S_3, (g) superimposed S_1, S_2, and S_3, and (h) the original image (40×40 "Lena" obtained by nearest color binary approximation).

An optimal n out of n visual cryptography scheme can be constructed by using the following two $n \times 2^{n-1}$ matrix collections:

$$C_0 = \{\pi(B_0)\}, \text{ and } C_1 = \{\pi(B_1)\},$$

where B_0 is a matrix whose columns are all binary vectors of size n with even number of ones, B_1 is a a matrix whose columns are all binary vectors of size n with odd number of ones, and $\pi(M)$ represents the collection of all matrices obtained by permuting the columns of a matrix M.

Here, the dealer encodes the pixel b into 2^{n-1} subpixels in each of the n shares. The bitwise OR of all rows of a matrix from C_0 contains a white subpixel. On the other hand, the bitwise OR of all rows of a matrix from C_1 does not contain a white subpixel. This feature distinguishes the white pixel from the black in the decoding process by the human visual system. Any $k < n$ shares do not reveal any information whatsoever about the secret image. In [84], Naor and Shamir proposed this (n, n)-VCS construction and proved that it is optimal. Figure 8.4 shows a (3, 3)-VCS applied to "Lena".

8.2.4 Constructing a (k, n)-VCS

The first few general k out of n visual cryptography schemes were introduced by Naor and Shamir in [84]. However, their constructions were quite inefficient, since a large number of subpixels were needed to encode a given pixel. A much more efficient construction mechanism was introduced by Droste in [85]. Droste's construction significantly reduces the number of subpixels needed for encoding.

Before we start discussing the algorithm, a few crucial definitions are presented. For a given $n \times m$ matrix M, a *k-restriction* of M is a $k \times m$ submatrix of M consisting of some k rows of M. Clearly, there are a total of $\binom{n}{k}$ k-restrictions of M. Boolean $n \times m$ matrices B_0 and B_1 are called *basis matrices* of a (k, n)-VCS if the two collections $C_0 = \{\pi(B_0)\}$ and $C_1 = \{\pi(B_1)\}$ constitute that (k, n)-VCS. Here, $\pi(M)$ represents the collection of all matrices obtained by permuting the columns of a matrix M. Within Droste's algorithm, a k-restriction of B_0 always contains all even Boolean vectors of size n among its columns, and a k-restriction of B_1 always contains all odd Boolean vectors of size n among its columns.

Besides these columns, a k-restriction of B_0 or B_1 could contain additional columns, which are referred to as the *remaining columns*. In Droste's construction, the remaining columns of any two k-restrictions of matrix B_i are always equal. Finally, the remaining columns of every k-restriction B_i that are not the remaining columns of every k-restriction of B_j, where $i \neq j$, are called the *rest* of B_i.

Now that we are equipped with all the necessary definitions, we proceed with the details Droste's Algorithm, which is known to always successfully terminate [85].

Droste's Algorithm

INPUT: Integers k and n
OUTPUT: Matrices B_0 and B_1, which are the basis matrices for a (k, n)-VCS

1. For all even $p \in \{0,\ldots,k\}$, do the following:
   a. If $p \leq k - p$, add all Boolean vectors of size n with p ones to B_0 as the new columns
   b. If $p > k - p$, add all Boolean vectors of size n with $(p + n - k)$ ones to B_0 as the new columns
2. For all odd $p \in \{0,\ldots,k\}$, do the following:
   a. If $p \leq k - p$, add all Boolean vectors of size n with p ones to B_1 as the new columns
   b. If $p > k - p$, add all Boolean vectors of size n with $(p + n - k)$ ones to B_0 as the new columns
3. While the rests of B_0 and B_1 are not empty do the following:
   a. Add to B_0 as new columns a minimal set of Boolean vectors of size n that eliminates the rest of B_1 (NOTE: This may expand the current rest of B_1)
   b. Add to B_1 as new columns a minimal set of Boolean vectors of size n that eliminates the rest of B_0 (NOTE: This may expand the current rest of B_0)

Next, we analyze the steps of the Droste's Algorithm by means of an example. Let us take $k = 4$ and $n = 5$. After the first step, the columns of matrix B_0 are all Boolean vectors of weight zero, two, and five:

$$B_0 = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

After the second step of the algorithm, the columns matrix B_1 are all Boolean vectors of weight one, and four:

$$B_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

In the first run of step 3, the algorithm updates B_0 and B_1 according to their rests. First, the rest of B_1 is determined. In this case, the rest of B_1 consists of vectors (0 0 0 0) and (1 1 1 1). Therefore, the minimal set of Boolean vectors of size $n = 5$ that is added to B_0 is represented by the columns of following matrix:

$$\begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

Once B_0 expands, its rest determines the expansion of B_1. The rest if B_0 consists of vectors all Boolean vectors of size k = 4 whose weight is 1. Thus, the minimal set of Boolean vectors of size $n = 5$ that is added to B_0 is represented by the columns of following matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Hence, at the end of the first run of step 3, basis matrices B_0 and B_1 have the following values:

$$B_0 = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix},$$

$$B_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

In the second and final run of step 3, the algorithm updates B_0 and the two rests become empty. The rest of B_1 is (0 0 0 0) so B_1 gets expanded by the column (0 0 0 0 0), and the final set of basis matrices for (4, 5)-VCS is:

$$B_0 = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{bmatrix},$$

$$B_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Similarly, by using the same algorithm, one obtains the following basis matrices for (3, 4)-VCS:

$$B_0 = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \text{ and } B_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

In Figure 8.5, the above two matrices are the basis matrices of two collections C_0 and C_1 used in the 3 out of 4 sharing of "Lena". C_0 is a collection of all matrices obtained by permuting the columns of B_0, while C_1 is a collection of all matrices obtained by permuting the columns of B_1. Observe from Figure 8.5 that a more significant loss of contrast occurs as the scheme becomes more complex (3 out of 4), however, the image is still visually recoverable.

Different, but also efficient constructions of (k, n)-VCS were proposed by Ateniese, et al, in [86]. Next, we cover *audio cryptography*, which uses similar mechanisms to the ones that we have just presented.

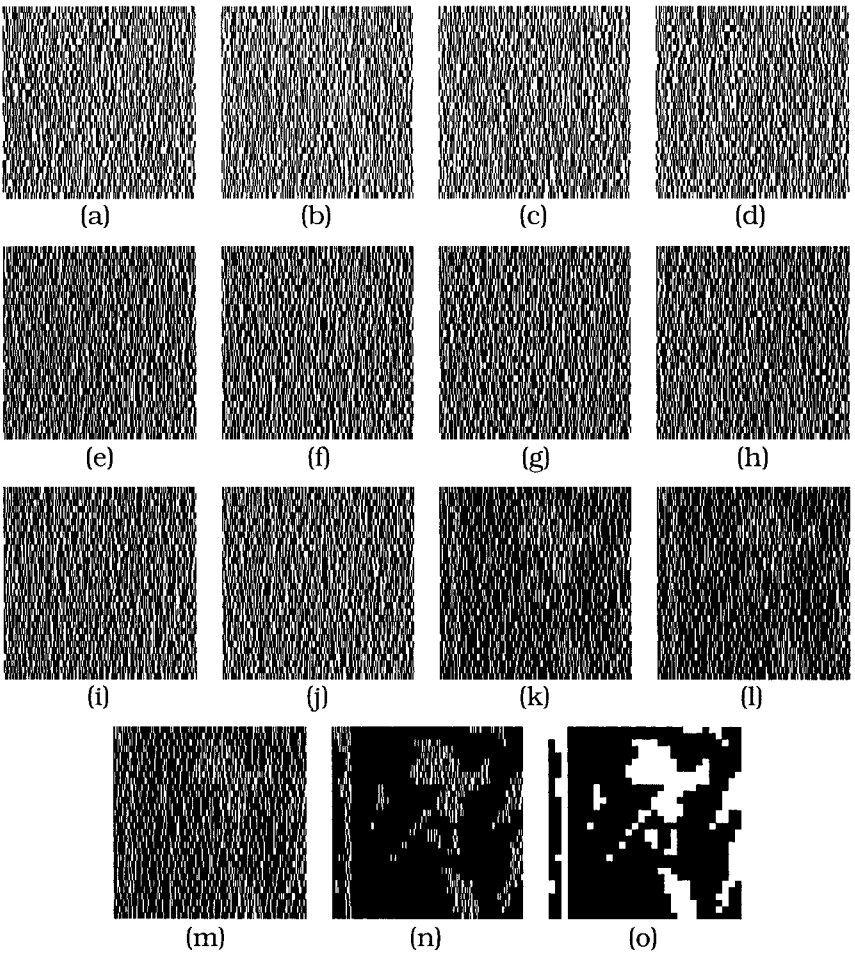


Figure 8.5. A (3, 4)-VCS of binary "Lena": (a) the first share S_1, (b) the second share S_2, (c) the third share S_3, (d) the fourth share S_4, (e) superimposed S_1 and S_2, (f) superimposed S_1 and S_3, (g) superimposed S_1 and S_4, (h) superimposed S_2 and S_3, (i) superimposed S_2 and S_4, (j) superimposed S_3 and S_4, (k) superimposed S_1, S_2, and S_3, (l) superimposed S_1, S_2, and S_4, (m) superimposed S_2, S_3, and S_4, (n) superimposed S_1, S_2, S_3, and S_4, and (o) the original image (30×30 "Lena" obtained by nearest color binary approximation).

8.3 AUDIO CRYPTOGRAPHY

There are many reasons for studying cryptographic schemes that do not rely on computers or other hardware to perform encryption or decryption [88]. As we saw in the previous section, visual cryptography studies cryptographic schemes that rely on the human visual system to reveal secret messages. Similarly, *audio cryptography* relies on the human auditory system to decrypt messages. For people with visual disabilities this is obviously a preferred scheme.

8.3.1 DHQ Audio Secret Sharing Scheme

In 1998, Desmedt, Hou, and Quisquater [88] proposed the first audio secret sharing (ASS) scheme, denoted by *DHQ ASS scheme.* The authors proposed constructions for a (2, 2) DHQ ASS scheme, and its extension to a more general (2, n) DHQ ASS scheme. A DHQ ASS scheme essentially embeds a binary secret message by a cover sound (harmonic sound or high quality music). The human auditory system can decode the secret message when any of the two shares are played simultaneously. A (2, 2) DHQ ASS uses one cover sound, but in the general case, a (2, n) DHQ ASS uses $\lceil \log_2 n \rceil$ different cover sounds.

Interference of Sound and Stereo Conception.

Sound interference occurs when the two sound waves meet while travelling through the medium (such as air or water). A sound wave is essentially a sequence of high-pressure and low-pressure parts. If the high-pressure part of one wave lines up with a low-pressure part of another wave during the interference, the pressure fluctuation becomes insignificant, which results in the low audio volume. This is referred to as the *destructive interference* (see Figure 8.6-a). On the other hand, if the high-pressure part of one wave interferes with a high-pressure part of another wave, the high-pressure of the resulting sound wave is intensified, and the audio volume increases. This is known as the *constructive interference* (see Figure 8.6-b). In order to achieve better interference, matching must occur in both space and time. This principle forms a basis for one of the proposed DHQ ASS decryption methods [88].

Figure 8.6-a shows the idealistic destructive interference of the two simple harmonic sounds that are of the same frequency and amplitude. The two sounds interfere with a 180 difference in phase, thus destroying each other. In Figure 8.6-b, the same sounds interfere in 0 difference in phase, and the amplitude of a resulting wave doubles.

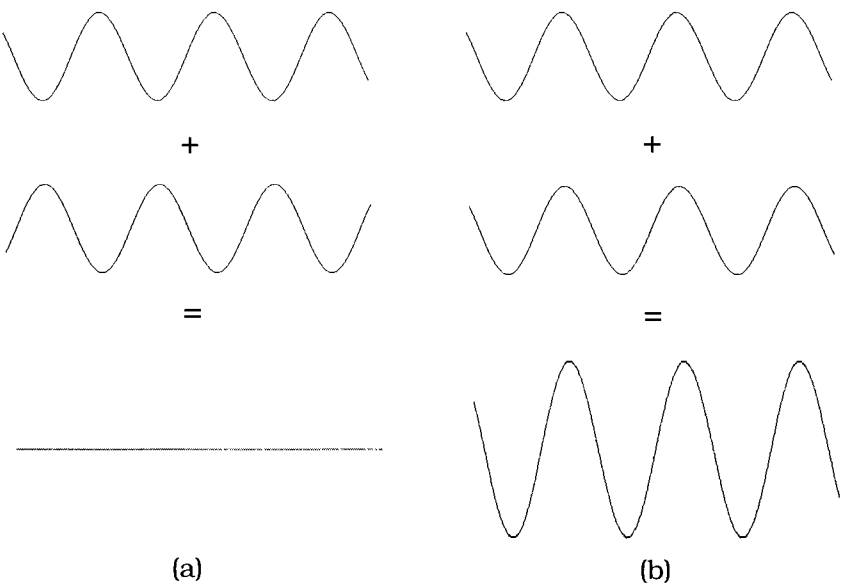


Figure 8.6. An illustration of sound interference for simple harmonic sounds: (a) destructive interference, and (b) constructive interference.

In [88], the authors proposed another decryption mechanism that is based on a similar but different principle called *stereo conception.* Namely, when the sound waves have different phases, the human auditory system is capable of detecting the direction from where the sounds have originated. Thus, if one sound source is directed towards the left ear, and the other towards the right ear, humans can detect the sound sources when the waves have different phases. However, when the two sound waves have identical phases, the human auditory system is deceived, and the sound appears as if it comes from the center, rather than from the sides. This property of

human auditory system is used in one of the DHQ ASS decryption methods.

8.3.2 A (2, 2) DHQ ASS Scheme

As mentioned earlier, a 2 out of 2 DHQ ASS scheme requires one cover sound. The cover sound is not restricted to harmonic sound, and a much more complex type of sound (such as high-quality music) could be used as the cover sound. In fact, better perceptual results were observed for schemes that used music as a cover sound, as opposed to the schemes that used harmonic cover sounds [88]. Next, we present an algorithm for generating a (2, 2) DHQ ASS scheme.

(2, 2) DHQ Encoding Algorithm

INPUT: Binary string S (the secret message), integer L (the length of S), floating-point number T (the number of seconds of sound used for encoding of one bit from S), and B (the cover sound that is $T \times L$ seconds long).

OUTPUT: Audio shares s_1 and s_2 that constitute a (2, 2) DHQ ASS scheme.

5. Initialize both s_1 and s_2 to B
6. For i from 1 to L do the following:
   a) Set σ_1 to be the T seconds long sound segment of s_1 starting from time $T \times (i - 1)$
   b) Set σ_2 to be the T seconds long sound segment of s_2 starting from time $T \times (i - 1)$
   c) Chose a random bit b
   d) Compute $b' = \neg(b \oplus S_i)$, where S_i is the i^{th} bit of S and $\neg$ is the standard logical negation
   e) If b is 1, multiply σ_1 with -1, implying a 180 (π) phase change
   f) If b' is 1, multiply σ_2 with -1, implying a 180 (π) phase change

Plug back the sound segments σ_1 and σ_2 to their proper place within s_1 and s_2 respectively.

Desmedt, Hou, and Quisquater proposed two decryption methods. One is based on the concept of sound interference, while the other is based on the concept of stereo conception. In the first method, two speakers are put very close while facing each other. One speaker plays the sound share s_1, and simultaneously the other speaker plays the sound share s_2. This results in sound interference and the listener can clearly observe the change in volume. Namely, the lower volume segment represents secret bit 0, and the louder volume segment represents secret bit 1. Ideally, a lower volume segment should be completely silent, however, due to many environmental factors the ideal destructive interference is nearly impossible to accomplish. In the second method, one speaker is placed to the left side of a listener and the other speaker is placed to the right side of a listener, both facing the listener's ears. Again the two shares are simultaneously played. Due to stereo conception, the listener observes changes in direction from where the sound originates. If the listener observes that the sound segment is coming from the sides, the two signals are out of phase, which means that the segment represents secret bit 0. Otherwise, the listener observes that the sound is coming from the center, which means that the two sounds have the same phase representing the secret bit 1. As a useful variant of this method, the headphones instead of two speakers could be used.

8.3.3 A (2, *n*) DHQ ASS Scheme

A more general (2, *n*) DHQ ASS scheme proposed in [88] uses $\lceil \log_2 n \rceil$ cover sounds. For practical purposes, the cover sounds could be different. For each of the $\lceil \log_2 n \rceil$ cover sounds one creates shares s_0^i and s_1^i ($1 \le i \le \lceil \log_2 n \rceil$) using the (2, 2) DHQ Encoding Algorithm. A participant j ($0 \le j \le n - 1$) receives the audio share s_0^i if the i^{th} bit of the binary representation of the integer j is zero. Otherwise, a participant j receives s_1^i.

The drawback of this scheme is that it needs $\lceil \log_2 n \rceil$ cover sounds. In 2003, Lin, Laih and Yang improved this (2, *n*) DHQ ASS scheme by using the time segment division [89]. The two improved (2, *n*) ASS schemes use only one cover sound. However, until recently, *n* out of *n*, *k* out of *n*, and general access structure audio sharing schemes were not known to exist. In [87], Socek and Magliveras have proposed efficient construction mechanisms for these most general audio sharing schemes.

8.4 AUDIOVISUAL CRYPTOGRAPHY

There exist a model by which one can generalize the constructions for audio and visual cryptography and use them to generate both audio and visual secret sharing schemes. This very nice generalization makes it possible to create the constructions for audio secret sharing schemes that were not yet proposed.

8.4.1 Socek-Magliveras Audio Cryptography Schemes

The proposed schemes by Socek and Magliveras are based on a different principle than that of Desmedt, Hou, and Quisquater. Instead of using sound interference, two types of flat frequencies (beeps) are used: the short beep and the long beep. This kind of sound is similar to a Morse code signal. An audio sharing scheme based on this principle is referred to as an *audio cryptography scheme* (ACS), and it resembles a strong analogy with the well-studied visual cryptography.

The ACS Model

In any Morse code audio message, there are two types of sounds: the short beep and the long beep. When making an analogy between audio and visual cryptography, the short beep corresponds to a white pixel, whereas the long beep corresponds to a black pixel. Mathematically, we represent the short beep by 0 and the long beep by 1. We also make use of the visually logical representation of beeps where - (the dash) denotes a long beep, and . (the dot) denotes a short beep. Unfortunately, Morse code is not a prefix code, and pauses are needed to distinguish between symbols. Since the goal is to minimize the amount of information needed for encoding, Morse coded audio messages are consequently not the best choice. Instead, a prefix code, possibly a Huffman encoding scheme based on the probability distribution of the symbols should be used. Thus, an audio signal represented in the long-beep short-beep fashion is hereby referred to as the *prefix binary code* (PBC) audio signal. One can think of a PBC audio signal as a simple binary message.

The basic idea of the schemes proposed in [87] can be best described by considering a (2,2)-ACS case. Let us consider a PBC audio message S, which contains exactly m beeps. The dealer creates two audio shares (binary sequences), S_1 and S_2, consisting of exactly 2m

beeps. A beep b from S is expanded into two beeps in each of the two shares. A short beep b is encoded with the same scheme in both shares, i.e, either $S_1(b) = S_2(b) =$.- or $S_1(b) = S_2(b) =$ -., depending on a coin flip. This will ensure that the simultaneous playback of the two shares contains a short beep. On the other hand, a long beep b is encoded with opposite schemes in the two shares. That is, either $S_1(b) =$._ and $S_2(b) =$ _., or $S_1(b) =$ _. and $S_2(b) =$._, depending on a coin flip. The effect of this is that simultaneously playing the two shares produces two long beeps. Thus, the superposition of a short beep with a second short beep in a particular time frame is perceived as a short beep, while the superposition of two beeps, at least one of which is long, is perceived as a long beep. Figure 8.7 shows the encoding of a beep in a (2, 2)-ACS.

	Coin toss = Head			Coin toss = Tail		
Short Beep	Share #1	Share #2	Superposed	Share #1	Share #2	Superposed
•	— •	— •	— •	• —	• —	• —
Long Beep	Share #1	Share #2	Superposed	Share #1	Share #2	Superposed
—	— •	• —	— —	• —	— •	— —

Figure 8.7. The encoding of a beep in (2, 2)-ACS: the dealer randomly tosses a coin, which determines the encoding scheme.

For example, let the secret message be ...___.... The dealer tosses a coin 9 times and the result turns out to be THHTHHHTH. Then, the corresponding two shares are:

._ _. _. ._ _. _. _. ._ _. and ._ _. _. _. ._ ._ _. ._ _.. When played simultaneously through speakers, the two shares produce the following sound: ._ _. _. __ __ __ _. ._ _.. Clearly, this reveals the secret plaintext, while observing the two shares individually gives no information about it. Note that the resulting sound corresponds to the bitwise OR of the two binary shares.

Using this principle, it has been shown that the 2 out of 2, the 2 out of n, the n out of n, and the k out of n visual cryptography constructions that are previously presented in this chapter are also

valid constructions for the 2 out of 2, the 2 out of n, the n out of n, and the k out of n audio cryptography schemes, respectively [87]. Furthermore, the authors showed that the constructions for general access structure visual cryptography proposed by Ateniese, et al. [86], can be applied to create new general access structure audio cryptography. Thus, one can think of the *audiovisual cryptography* as the more general discipline that studies secret sharing schemes applicable to both visual and audio cryptography.

SUMMARY

Secret sharing in general deals with sharing a secret number, which has applications in secure key management and multiparty secure protocols. Visual and audio cryptography did not find the true application yet, however, due to many contributions over the last decade, we now have optimal and near-optimal constructions for all possible visual and audio cryptography schemes.

REFERENCES

[1] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", *IETF RFC 3711*, March 2004, [http://www.ietf.org/rfc/rfc3711.txt].

[2] P.C. van Oorschot, A.J. Menezes, and S.A. Vanstone, "Handbook of Applied Cryptography", *CRC Press, Inc.*, 1997.

[3] B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C", Second Edition, *John Wiley & Sons*, 1996.

[4] J. Pieprzyk, T. Hardjono, and J. Seberry, "Fundamentals of Computer Security", *Springer-Verlag*, 2003.

[5] D.R. Stinson. "Cryptography: Theory and Practice", Second Edition, *CRC Press*, 2002.

[6] B. Furht, D. Socek, and A.M. Eskicioglu, "Multimedia Security Handbook", ed. by B. Furht and D. Kirovski, *volume 4 of Internet and Communications Series*, chapter "Fundamentals of Multimedia Encryption Techniques". *CRC Press*, December 2004.

[7] Shujun Li, Guanrong Chen and Xuan Zheng. "Multimedia Security Handbook", ed. by B. Furht and D. Kirovski, *volume 4 of Internet and Communications Series*, chapter "Chaos-Based Encryption for Digital Images and Videos". *CRC Press*, December 2004.

[8] B. Furht and D. Socek, "Multimedia security: Encryption techniques", in *Network Security: Technology Advances, Strategies, and Change Drivers, IEC Comprehensive Report, International Engineering Consortium*, Chicago, IL, 335-349, 2004.

[9] FIPS 46, "Data Encryption Standard (DES)", Federal Processing Standards Application 46, *U.S. Department of Commerce/ National Bureau of Standards, National Technical Information Service*, Springfield, Virginia, 1977 (revised as FIPS 46-1 in 1988, FIPS 46-2 in 1993, and FIPS 46-3 in 1999).

[10] J.L. Massey and X. Lai, "Device for the Conversion of a Digital Block and Use of Same", *US Patent #5,214,703*, 25 May 1993.

[11] FIPS 197, "Advanced Encryption Standard (AES)", Federal Processing Standards Application 197, *U.S. Department of Commerce/U.S. National Institute of Standards and Technology (NIST)*, 2001.

[12] H. Feistel, "Block Cipher Cryptographic System", *US Patent #3,798,359*, 19 March 1974.

[13] W. Diffie, and M.E. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, IT-22(6):644-654, 1976.

[14] D. Kahn, "The Codebreakers, The Story of Secret Writing", *Macmillan Publishing Co.*, New York, 1967.

[15] S. Singh, "The Code Book: The Secret History of Codes and Code-breaking", *Forth Estate Ltd.*, 1999.

[16] X. Zou, B. Ramamurthy, and S.S. Magliveras, "Secure Group Communications Over Data Networks", *Springer-Verlag*, 2004.

[17] R.L. Rivest, A. Shamir, and L.M. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems", *Communications of the ACM*, 21(2):120-126, 1978.

[18] R.L. Rivest, A. Shamir, and L.M. Adleman, "Cryptographic Communications System and Method", *US Patent #4,405,829*, 20 September 1983.

[19] T. ElGamal, "A Public-key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Transactions on Information Theory*, 31(4):469-472, 1985.

[20] N. Koblitz, "Elliptic Curve Cryptosystems", *Mathematics of Computation*, 48:203-209, 1987.

[21] V.S. Miller, "Use of Elliptic Curves in Cryptography", Advances in Cryptology, CRYPTO '85, *Lecture Notes in Computer Science, Springer-Verlag*, 218:417-426, 1986.

[22] J. Hoffstein, J. Pipher, and J.H. Silverman, "NTRU: A Ring Based Public-key Cryptosystem", in Proc. *Algorithmic Number Theory (ANTS III), Lecture Notes in Computer Science, Springer-Verlag*, 1423:267-288, 1998.

[23] N. Howgrave-Graham, P. Nguyen, D. Pointcheval, J. Proos, J.H. Silverman, A. Singer, and W. Whyte, "The Impact of Decryption Failures on the Security of NTRU Encryption", in Proc. CRYPTO '03, *Lecture Notes in Computer Science, Springer-Verlag*, 2729:226-246, 2003.

[24] T. Seidel, D. Socek, and M. Sramka, "Parallel Symmetric Attack on NTRU using Non-Deterministic Lattice Reduction", *Designs, Codes and Cryptography, Kluwer Academic Publishers*, 32(1-3): 369-379, 2004.

[25] G.S. Vernam, "Secret Signalling System", *US Patent #1,310,719*, 22 July 1919.

[26] FIPS 180, "Secure Hash Standard (SHS)", Federal Processing Standards Application 180, *U.S. Department of Commerce/U.S. National Institute of Standards and Technology (NIST)*, 1993 (revised as FIPS 180-1 in 1995, FIPS 180-2 in 2002).

[27] FIPS 186, "Digital Signature Standard (DSS)", Federal Processing Standards Application 186, *U.S. Department of Commerce/U.S. National Institute of Standards and Technology (NIST)*, 1994 (revised as FIPS 186-1 in 1998, FIPS 186-2 in 2000 and 2001).

[28] A. Kerckhoffs, "La Cryptographie Militaire", *Journal des Sciences Militaires*, 9:161-191, 1883.

[29] J. Daemen, V. Rijmen, "The Block Cipher Rijndael", Smart Card Research and Applications, *Lecture Notes in Computer Science, Springer-Verlag*, 1820: 288-296, 2000.

[30] M.O. Rabin, "Probabilistic Algorithm for Primality Testing", *Journal of Number Theory*, 12:128-138, 1980.

[31] M. Van Droogenbroeck and R. Benedett, "Techniques for a Selective Encryption of Uncompressed and Compressed Images", in Proc. *Advanced Concepts for Intelligent Vision Systems (ACIVS 2002)*, Ghent, Belgium, September 9-11, 90-97, 2002.

[32] M. Podesser, H.-P. Schmidt and A. Uhl, "Selective Bitplane Encryption for Secure Transmission of Image Data in Mobile Environments", in Proc. *Fifth IEEE Nordic Signal Processing Symposium (NORSIG 2002)*, Tromso-Trondheim, Norway, October 4-7, 2002.

[33] L. Tang, "Methods for Encrypting and Decrypting MPEG Video Data Efficiently", in Proc. *Fourth ACM International Multimedia Conference*, Boston, MA, November 18-22, 219-230, 1996.

[34] L. Qiao, K. Nahrstedt, and M.-C. Tam, "Is MPEG Encryption by Using Random List Instead of Zigzag Order Secure?", in Proc. *IEEE International Symposium on Consumer Electronics*, Singapore, December 2-4, IEEE Computer Society, 226-229, 1997.

[35] L. Qiao and K. Nahrstedt, "Comparison of MPEG Encryption Algorithms", *International Journal on Computer and Graphics*, Special Issue on Data Security in Image Communication and Network, 22(3):437-448, 1998.

[36] C. Shi and B. Bhargava, "A Fast MPEG Video Encryption Algorithm", in Proc. *Sixth ACM International Multimedia Conference*, Bristol, UK, September 12-16, 1998, ACM Electronic Proceedings [http://turing.acm.org/sigs/sigmm/MM98/electronic_proceedings/shi/].

[37] C. Shi, S.-Y. Wang and B. Bhargava, "MPEG Video Encryption in Real-Time Using Secret Key Cryptography", in Proc. *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '99)*, Las Vegas, NV, June 28 - July 1, CSREA Press, 6:2822-2828, 1999.

[38] B. Bhargava, C. Shi, and S.-Y. Wang, "MPEG Video Encryption Algorithms", *Multimedia Tools and Applications, Kluwer Academic Publishers*, 24(1):57-79, 2004.

[39] T.E. Seidel, D. Socek, and M. Sramka, "Cryptanalysis of Video Encryption Algorithms", in Proc. *Third Central European Conference on Cryptology (TATRACRYPT 2003)*, Bratislava, Slovak Republic, June 26-28 (2003), *Tatra Mountains Mathematical Publications*, 29:1-9, 2004.

[40] H. Cheng and X. Li, "Partial Encryption of Compressed Images and Video", *IEEE Transactions on Signal Processing*, 48(8): 2439-2451, 2000.

[41] J.M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients", *IEEE Transactions on Signal Processing*, 41(12):3445-3462, 1993.

[42] A. Said W.A. Pearlman, "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees", *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243-250, 1996.

[43] J.W. Woods, "Subband Image Coding", *Kluwer Academic Publishers*, Norwell, MA, 1990.

[44] J.-C. Yen and J.-I. Guo, "A New Chaotic Key-Based Design for Image Encryption and Decryption", in Proc. *IEEE International Conference on Circuits and Systems (ISACS 2000)*, 4:49-52, 2000.

[45] L.-G. Jiang, H.-W. Zhu, D. He, C. He, and G.-R. Hu, "Chaotic Characteristics of a One-Dimensional Iterative Map with Infinite Collapses", *IEEE Transactions on Circuits and Systems: Fundamental Theory and Applications*, 48:900-906, 2001.

[46] S. Li, G. Chen and X. Mou, "On the Dynamical Degradation of Digital Piecewise Linear Chaotic Maps", accepted by *International Journal of Bifurcation and Chaos* in August 2004, [http://www.hooklee.com/Papers/IJBC2004.pdf].

[47] S. Li and X. Zheng, "Cryptanalysis of a Chaotic Image Encryption Method", in Proc. *IEEE International Symposium on Circuits and Systems (ISCAS 2002)*, 2:708-711, 2002.

[48] H.-C. Chen and J.-C. Yen, "A New Cryptography System and Its VLSI Realization", *Journal of Systems Architechture*, 49(7-9):355-367, 2003.

[49] H.-C. Chen, J.-C. Yen, and J.-I. Guo, "Design of a New Cryptography System", in Proc. *Advances in Multimedia Information Processing (PCM 2002): Third IEEE Pacific Rim Conference on Multimedia Processing, Lecture Notes in Computer Science*, 2532:1041-1048, 2002.

[50] S. Li, C. Li, G. Chen, and X. Mou, "Cryptanalysis of the RCES/RSES Image Encryption Scheme", submitted to *IEEE Transactions on Image Processing* at Mar. 9, 2004, [http://eprint.iacr.org/2004/376.pdf].

[51] J. Fridrich, "Image Encryption Based on Chaotic Maps", in Proc. *IEEE International Conference on Sysytems, Man and Cybernetics (ICSMC'97)*, October 12-15, 2:1105-1110, 1997.

[52] F. Pichler and J. Scharinger, "Ciphering by Bernoulli-Shifts in Finite Abelian Groups", In *Contributions to General Algebra 9*, eds. H.K. Kaiser, W.B. Müller and G.F. Pilz, 249-256, 1995.

[53] J. Scharinger and F. Pichler, "Efficient Image Encryption Based on Chaotic Maps", in Proc. *20th Workshop of the AAPR, Pattern Recognition*, Oldenbourg Verlag, 159-170, 1996.

[54] M. Salleh, S. Ibrahim, and I.F. Isinn, "Ciphering Key of Chaos Image Encryption", in Proc. *International Conference on Artificial Intelligence in Engineering and Technology (ICAIET 2002)*, 58-62, 2002.

[55] M. Salleh, S. Ibrahim, and I.F. Isinn, "Enhanced Chaotic Image Encryption Algorithm Based on Baker's Map", in Proc. *IEEE International Symposium on Circuits and Systems (ISCAS 2003)*, II:508-511, 2003.

[56] G. Chen, Y. Mao, and C.K. Chui, "A Symmetric Image Encryption Scheme Based on 3D Chaotic Cat Maps", *Chaos, Solitons & Fractals*, 21(3):749-761, 2004.

[57] G. Chen, T. Ueta, "Yet Another Chaotic Attractor", *International Journal of Bifurcation and Chaos*, 9(7):1465-1466, 1999.

[58] T. Ueta, G. Chen, "Bifurcation Analysis of Chen's Equation", *International Journal of Bifurcation and Chaos*, 10(8):1917-1931, 2000.

[59] M. Van Droogenbroeck, "Partial Encryption of Images for Real-Time Applications", in Proc. *Fourth IEEE Benelux Signal Processing*, Hilvarenbeek, The Netherlands, April 15-16, 11-15, 2004.

[60] W. Tuchman, "Hellman Presents No Shortcut Solutions to DES", *IEEE Spectrum*, 16(7):40-41, July 1979.

[61] J. Fridrich, M. Goljan, and R. Du, "Lossless Data Embedding for All Image Formats", in Proc. *SPIE Photonic West, Electronic Imaging, Security and Watermarking of Multimedia Contents*, San Jose, California, 4675:572-583, January 2002.

[62] T. Lookabaugh, D. C. Sicker, D. M. Keaton, W. Y. Guo and I. Vedula, "Security Analysis of Selectively Encrypted MPEG-2 Streams", *Multimedia Systems and Applications VI Conference*, Orlando, FL, September 7-11, 2003.

[63] Peter Symes, "Digital Video Compression", *McGraw-Hill*, New York, NY, 2004.

[64] T. Ebrahimi and F. Pereira, "The MPEG-4 Book", *Prentice Hall*, Upper Saddle River, New Jersey, 2002.

[65] J. Meyer and F. Gadegast, "Security Mechanisms for Multimedia Data with the Example MPEG-1 Video", Project Description of SECMPEG, *Technical University of Berlin*, Germany, May 1995, [http://www.gadegast.de/frank/doc/secmeng.pdf].

[66] G.A. Spanos and T.B. Maples, "Performance Study of Selective Encryption Scheme for the Security of Networked Real-Time Video", in Proc. *Fourth International Conference on Computer Communications and Networks (ICCCN '95)*, Las Vegas, NV, September 20-23, 2-10, 1995.

[67] G.A. Spanos and T.B. Maples, "Security for Real-Time MPEG Compressed Video in Distributed Multimedia Applications", in Proc. *IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications*, Scottsdale, AZ, March 27-29, 72-78, 1996.

[68] I. Agi and L. Gong, "An Empirical Study of Secure MPEG Video Transmission", in Proc. *IEEE Symposium on Network and Distributed System Security (SNDSS '96)*, San Diego, CA, February 22-23, IEEE Computer Society, 137-144, 1996.

[69] L. Qiao and K. Nahrstedt, "A New Algorithm for MPEG Video Encryption", in Proc. *First International Conference on Imaging Science, Systems and Technology (CISST '97)*, Las Vegas, NV, June 30-July 3, 21-29, 1997.

[70] A.M. Alattar and G.I. Al-Regib, "Evaluation of Selective Encryption Techniques for Secure Transmission of MPEG Video Bit-Streams", in Proc. *IEEE International Symposium on Circuits and Systems*, 4:340-343, 1999.

[71] A.M. Alattar, G.I. Al-Regib and S.A. Al-Semari, "Improved Selective Encryption techniques for Secure Transmission of MPEG Video Bit-Streams", in Proc. *International Conference on Image Processing (ICIP '99)*, Kobe, Japan, October 24-28, 4:256-260, 1999.

[72] C.-P. Wu and C.-C.J. Kuo, "Fast Encryption Methods for Audiovisual Data Confidentiality", in Proc. *SPIE International Symposia on Information Technologies*, Boston, MA, 4209:284-295, November 2000.

[73] C.-P. Wu and C.-C. J. Kuo, "Efficient Multimedia Encryption via Entropy Codec Design", in Proc. *SPIE Security and Watermarking of Multimedia Content*, San Jose, CA, 4314:128-138, January 2001.

[74] W. Zeng and S. Lei, "Efficient Frequency Domain Selective Scrambling of Digital Video", *IEEE Transactions on Multimedia*, 5(1):118-129, March 2002.

[75] ITU, "Recommendation G.723.1 Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 7.3 kb/s", *International Telecommunication Union*, 1996.

[76] A. Servetti and J.C. De Martin, "Perception Based Partial Encryption of Compressed Speech", *IEEE Transaction on Speech and Audio Processing*, 10(8):637-643, 2002.

[77] R. Salami, C. Laflamme, J.P. Adoul, A. Kataoka, S. Hayashi, T. Moriya, C. Lamblin, D. Massaloux, S. Proust, P. Kroon, and Y. Shoham, "Design and Description of CS-ACELP: A Toll Quality 8kb/s Speech Coder", *IEEE Transactions on Speech and Audio Processing*, 6(2):116-130, 1998.

[78] N.J. Thorwirth, P. Horvatic, R. Weis and J. Zhao, "Security Methods for MP3 Music Delivery", in Proc. *34th Asilomar Conference on Signals, Systems and Computers*, 2:1831-1835, 2000.

[79] A. Servetti, C. Testa and J. C. De Martin, "Frequency-Selective Partial Encryption of Compressed Audio", in Proc. *IEEE International Conference on Aucoustics, Sppech and Signal Processing*, Hong-Kong, April 6-10, 5:668-671, 2003.

[80] ISO/IES International Standard Information Technology, "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mb/s", *ISO/IEC 11172-3*, 1996.

[81] K. Brandenburg and G. Stoll, "ISO-MPEG-1 Audio: A Generic Standard for Coding of High-Quality Digital Audio", *Journal of the AES*, 4(10): 780-792, 1994.

[82] G.R. Blakley, "Safeguarding Cryptographic Keys", in Proc. *AFIPS National Computer Conference*, 48:313-317, 1979.

[83] A. Shamir, "How to Share a Secret", *Communications of the ACM*, 22:612-613, 1979.

[84] M. Naor and A. Shamir, "Visual Cryptography", in Proc. *Advances in Cryptology EUROCRYPT '94, Lecture Notes in Computer Science, Springer-Verlag*, 950:1-12, 1995.

[85] S. Droste, "New Results on Visual Cryptography", in Proc. *CRYPTO '96, Lecture Notes in Computer Science, Springer-Verlag*, 1109:401-415, 1996.

[86] G. Ateniese, C. Blundo, A. De Santis, and D.R. Stinson, "Constructions and Bounds for Visual Cryptography", in Proc. *23rd International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science, Springer-Verlag* 1099:416--428, 1996.

[87] D. Socek and S.S Magliveras, "General Access Structures are Possible in Audio Cryptography", submitted to *IEEE Transactions on Information Theory* at Jan. 26, 2005.

[88] Y. Desmedt, S. Hou, and J.-J. Quisquater, "Audio and Optical Cryptography", in Proc. *Advances in Cryptology - ASIACRYPT '98, Lecture Notes in Computer Science, Springer-Verlag*, 1514:392-404, 1998.

[89] C.-C. Lin, C.-S. Laih, and C.-N. Yang, "New Audio Secret Sharing Schemes With Time Division Technique", *Journal of Information Science and Engineering*, Institute of Information Science, Academia Sinica, Taipei, 19(4):605-614, 2003.

Part III

Multimedia Watermarking

Chapter 9

INTRODUCTION TO WATERMARKING

Recent proliferation and success of the Internet, together with the availability of relatively inexpensive digital recording and storage devices has created an environment in which it becomes very easy to obtain, replicate and distribute digital content without any loss in quality. This has become a great concern to the multimedia content (music, video, and image) publishing industries, because technologies or techniques to protect intellectual property rights for digital media and to prevent unauthorized copying did not exist.

Encryption technologies can be used to prevent unauthorized access to digital content. However, encryption has its limitations in protecting intellectual property rights because once digital content gets decrypted, there is nothing to prevent an authorized user from illegally replicating it. Another technology is obviously needed to help both establish and prove ownership rights, then track content usage, ensure authorized access, facilitate content authentication and prevent illegal replication.

This need attracted attention from the research community and industry leading to a creation of a new information hiding form, called *Digital Watermarking*. The basic idea of digital watermarking is to create a metadata containing information about the digital content to be protected, and then hide the metadata within that content. The information stored as metadata can have different formats, such as character string or binary image pattern, as illustrated in Figure 9.1. The metadata is first mapped into its bit stream representation, and then into a *watermark*, a pattern of the same type and dimension as the *cover work*, the digital content to be protected. The watermark is then embedded into the cover work where it should be imperceptible. The watermark should be robust enough to survive not only most common signal distortions but also distortions caused by malicious attacks.

It is clear that digital watermarking and encryption technologies are complementing each other, and that a complete multimedia security solution depends on both. This part of the book provides an overview of the image watermarking techniques. It starts with a description of various watermarking applications. Then it presents general concepts of digital watermarking, and digital watermarking techniques suitable for gray-level and color images.

We will see that many data hiding techniques have been developed for digital gray-scale and color images [35]. Those techniques typically make small modifications to the color or brightness values of the selected set of pixels without causing visually noticeable image distortion.

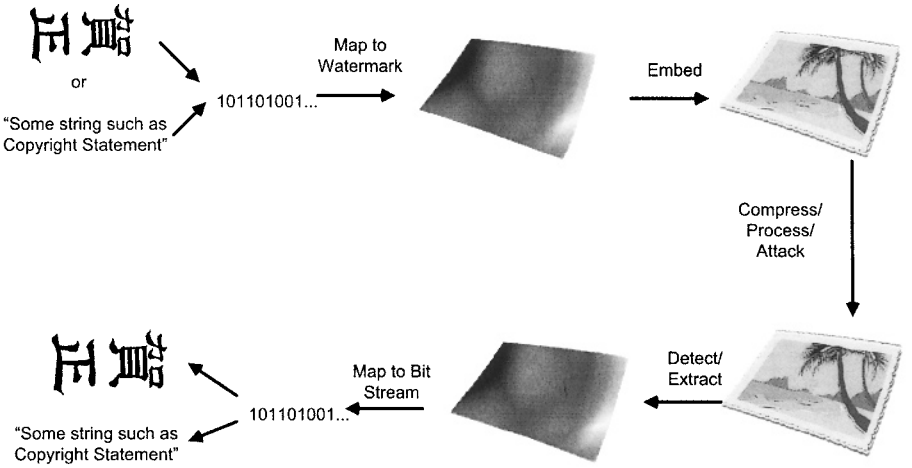


Figure 9.1. General framework of digital watermarking systems

The next chapter deals with issues related to watermarking of binary images. Binary images are a special kind of gray-level images. They have only one bit of gray-level resolution, which means that each binary image pixel has two possible gray-level values, '0' and '1'. Binary images can be created by scanning two-color documents, such as legal papers, birth certificates, digital books, engineering maps, music scores, etc. Since documents represent a primary form of written communication in our society, and large volumes are exchanged daily, document recipients should be able to authenticate documents as well as identify document owners. Digital watermarking can be specifically used for that purpose. However,

watermarking techniques designed for gray-scale and color images cannot be directly applied to the binary document images. It is not possible to make just a small modification of binary image pixel values, because binary image pixels have only two distinct values.

The next chapter introduces two-level watermarking. This is a digital watermarking technique that embeds two watermarks instead of one, and those two embedded watermarks serve different purposes. One watermark is embedded as a fragile mark, and it is used for image authentication and detection of image tampering. If the embedded watermark can be detected in the test image, the image is declared as an authentic one. Any modification of the watermarked image will render the watermark undetectable, indicating that the image was potentially tampered with. An authentication watermark embedded in an image can only be used to prove that an image was not altered. If the watermark detector fails to extract the embedded watermark, the only possible conclusion is that the test image is not authentic. However, failure to detect the fragile mark can have several causes. First, failure could be caused by illegal tampering with the intention to make a significant change in the conveyed meaning. This is the type of malicious image tampering an authentication watermarking scheme should identify. Failure could also be caused by standard image processing operations and manipulations, such as image compression, printing and scanning. Those modifications are not significant in the sense that they do not change the content and meaning of an image, and consequently, the test image is still an authentic one. Finally, failure could result from an original image that was not marked at all. This situation occurs when the original image simply belongs to someone else, and no malicious and illegal image tampering is involved at all. This problem can be solved by embedding the second watermark into an image. The second watermark is embedded as a robust mark, and it can be used to convey ownership information. This information can help identify the reason for an authentication test failure.

Chapter 10

APPLICATIONS OF DIGITAL WATERMARKING

Frequently there is a need to associate additional information with a digital content, such as music, image or video. For example, a copyright notice may need to be associated with an image to identify the legal owner, a serial number may need to be associated with a video to identify a legitimate user, or an identifier may need to be associated with a song to locate a database to obtain more information. This additional information can be linked with digital content by placing it in the header of a digital file, or for images, information can be encoded as a visible notice. Storing information in the header of a digital file has a couple of disadvantages. First, the information stored in the file header may not survive a file format conversion, and second, once an image is displayed or printed, its association with the information stored in the file header is lost. Adding a visible notice to an image may not be acceptable if it negatively affects the esthetics of the image. This can be corrected to some extent by making the notice as small as possible or by moving it to a visually insignificant portion of the image, such as the edge. However, once on the edge, this additional information can easily be cropped off, either intentionally or unintentionally.

This is exactly what happened with an image of Lena Soderberg after its copyright notice was cropped off (see Figure 10.1). The image was originally published as a Playboy centerfold in November 1972. After the image has been scanned for use as the test image, most of it has been cropped including the copyright notice printed on the edge of the image. The "Lena" image became the most frequently used test image in image processing research and appeared in a number of journal articles without any reference to its rightful owner, Playboy Enterprises, Inc.

Digital watermarking is an appropriate method for associating this additional information, the metadata, with a digital work. The

metadata is imperceptibly embedded as a watermark in a digital content, the cover work, and becomes inseparable from it.

Figure 10.1. The Lena image used as a test image on the left, and the cropped part of the original image, which identifies the copyright owner, Playboy Enterprises, Inc. on the right.

Furthermore, since watermarks will go through the same transformations as the cover work they are embedded within, it is sometimes possible to learn whether and how the content has been tampered with by looking into the resulting watermarks.

10.1 CLASSIFICATION OF WATERMARKING APPLICATIONS

There are a number of different watermarking application scenarios, and they can be classified in a number of different ways. The classification presented in Table 10.1 is based on the type of information conveyed by the watermark [57].

There's a large and important class of applications which use watermarks to provide some level of protection of intellectual property rights. The applications from that class use watermarks as carrier for information about content ownership and intellectual property rights. Those applications can be used for copyright protection, copy protection, and fingerprinting. Second class of applications uses watermarks for content verification. The applications from that class embed watermarks into multimedia digital content to ensure that the original content has not been altered, and to help determine the type and location of alteration in the case that the original content was modified. The third class of

applications uses watermarks to provide some additional information about the content. Broadcast monitoring and system enhancement applications belong to this class.

Table 10.1. Classes of watermarking applications.

Application Class	Watermark Purpose	Application Scenarios
Protection of Intellectual Property Rights	Conveys information about content ownership and intellectual property rights	• Copyright Protection • Copy Protection • Fingerprinting
Content Verification	Ensures that the original multimedia content has not been altered, and/or helps determine the type and location of alteration	• Authentication • Integrity Checking
Side-Channel Information	Represents side-channel used to carry additional information.	• Broadcast Monitoring • System Enhancement

We will provide a more detailed explanation of possible application scenarios involving watermarking in the next several sections.

10.2 COPYRIGHT PROTECTION

Copyright protection is one of the first applications targeted by digital watermarking. The metadata contains information about the copyright owner, and it is imperceptibly embedded as a watermark in the cover work to be protected. If users of digital content (music, images, and video) have easy access to watermark detectors, they should be able to recognize and interpret the embedded watermark and identify the copyright owner of the watermarked content.

An example of one commercial application created for that purpose is Digimarc Corporation's ImageBridge Solution. The ImageBridge watermark detector is made available in a form of plug-ins for many

popular image processing solutions, such as Adobe PhotoShop or Corel PhotoPaint. When a user opens an image using a Digimarc-enabled application, Digimarc's watermark detector will recognize a watermark. It will then contact a remote database using the watermark as a key to find a copyright owner and his contact information. An honest user can use that information to contact the copyright owner to request permission to use the image.

Having demonstrated earlier how an invisibly embedded watermark can be used to identify copyright ownership, it would be ideal if an embedded watermark could be used to prove the ownership as well, perhaps even in a court of law. Envision the following scenario: A copyright owner distributes his digital content with his invisibly embedded watermark. In the case of a copyright ownership dispute, a legal owner should be able to prove his ownership by demonstrating that he owns the original work, and that the disputed work has been derived from the original by embedding a watermark into it. This could be done by producing the original work together with the watermark detector, then using the detector to detect the owner's watermark in a disputed work. Unfortunately, it appears that the above scenario can be defeated under certain assumptions. Because of that, watermarking has not yet been accepted as a technology dependable enough to prove the ownership. One potential problem is related to the availability of a watermark detector. It has been demonstrated that if a detector is widely available, then it is not possible to protect watermark security. In other words, if a detector is available, it is always possible to remove an embedded watermark. This can be achieved by repeatedly making imperceptible changes to the watermarked work, until a watermark detector fails to detect the watermark. Once the watermark is removed, the original owner will no longer be able to prove his ownership. Even if the original watermark cannot be removed, Craver et al. [21][22][23] demonstrated that, under certain conditions, it is possible to watermark an already watermarked image in such a way as to make it appear that this second watermark is present in all copies of disputed image, including the original image. This can be achieved, for example, by subtracting the second watermark from a watermarked image, and using that resulting image as a new original image. The rightful owner saves and protects the original cover image C, and distributes the watermarked version of that image $C+W_1$, hoping that in a case of ownership dispute he will be able to prove that the $C+W_1$ was derived

from the original cover image C he owns. Unfortunately, the creator of the new fake original $C+W_1-W_2$ can make similar ownership claims. Assuming that W_1 and W_2 do not interfere with each other, the original cover image C can be shown to contain the watermark W_2, making it possible for the creator of the fake original to claim that the real original image C and its watermarked version $C+W_1$ were derived from the fake original $C+W_1-W_2$.

This is known as an ambiguity attack, and as the example above demonstrated, it can be used not only to dispute the ownership claims of the rightful copyright owner, but also to make a new ownership claim to the original digital content.

10.3 COPY PROTECTION

The objective of a copy protection application is to control access to and prevent illegal copying of a copyrighted content. It is an important application, especially for digital content, because digital copies can be easily made, they are perfect reproductions of the original, and they can be easily and inexpensively distributed over the Internet without quality degradation.

There are a number of technical and legal issues that need to be addressed and resolved in order to create a working copy protection solution. Those issues are difficult to resolve in open systems, and that is the reason why an open system copy protection solution has yet to be made. Copy protection is feasible in closed, proprietary systems, such as the Digital Versatile Disk (DVD) copy protection solution [7].

The DVD copy protection system has a number of components designed to provide copy protection at several levels. The Content Scrambling System (CSS) encrypts MPEG-2 video and makes it unusable to anyone who does not have a decoder and a pair of keys required to decrypt it. But, once the video has been decrypted, CSS does not provide any additional protection for the content.

Additional mechanisms have been put in place to provide extra protection for the decrypted (or unscrambled) video. For example, the Analog Protection System (APS) prevents an unscrambled video displayed on television from being recorded on an analog device, such as a VCR. APS does it by modifying the National Television

System Committee (NTSC) or Phase Alternating Line (PAL) signals in such a way that video can still be displayed on television but cannot be recorded on a VCR.

There was also a need to support limited copying of video content. A customer should be able to make a single copy of the broadcast video for later viewing, which is known as time shifting recording. However, a customer should not be able to make additional copies. The Copy Control Management System (CCMS) has been designed to provide that level of copy control by introducing and supporting three rules for copying: Copy-Free, Copy-Never, and Copy-Once. Two bits are needed to encode those rules, and the bits can be embedded into the video frames in the form of watermarks.

This copy control mechanism will work only if every DVD recorder contains a watermark detector. The problem is how to ensure that every DVD recorder has the watermark detector. There is no natural economic incentive for DVD manufacturers to increase the production cost of their product by incorporating watermark detectors in DVD recorders. After all, the perceived market value of a DVD recorder with a watermark detector may be lower compared with a recorder without it, because a customer would rather have a device that can make illegal copies.

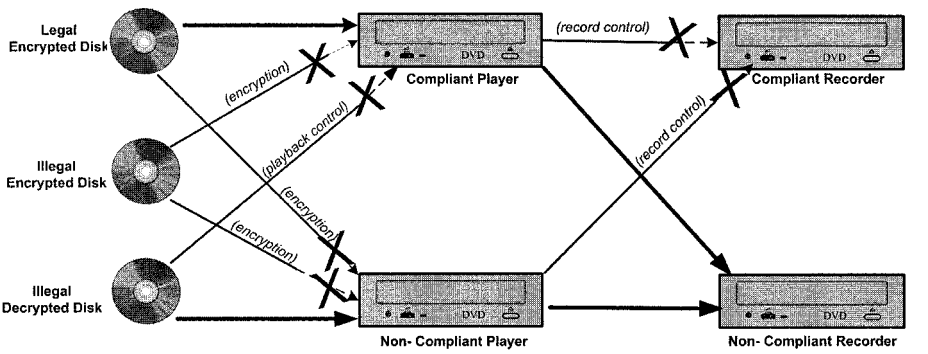


Figure 10.2. DVD copy protection system. The rules for copying are encoded using two bits which are embedded into the video frames in the form of watermarks.

This problem can be solved by forcing DVD manufacturers to add watermark detectors to their devices by law. Unfortunately, such a law does not exist. Even if it did, it would be very difficult to enforce

it across every country in the world. Therefore, an alternative solution is needed. The solution that has been adopted for DVD systems is based on the patent license. Basically, the DVD encryption patent license makes it mandatory to use watermark detectors in the patent compliant devices.

The patent license approach ensures that compliant devices will use watermark detectors and prevent illegal copying, but it also makes it legal to manufacture noncompliant devices. Devices that do not implement the patented decryption do not have to implement a watermark detector. Consequently, the DVD copy control mechanism does not prevent all possible illegal copying.

Interaction of encryption and copy control, combined with the playback control, a mechanism that allows a DVD compliant device to detect illegal copies, is used to create a solution where only illegal copies can be played on noncompliant devices, and only legal copies can be played on compliant devices. This is illustrated by Figure 10.2. The objective of this scheme is to insure that one device will not be able to play both legal and illegal content. If a customer wants to play both legal and illegal copies he will have to purchase both compliant and noncompliant devices. However, if one of the two devices has to be selected, the hope is that most customers will choose a compliant one.

10.4 FINGERPRINTING

There are some applications where the additional information associated with digital content should contain information about the end user, rather than about the owner of a digital content. Consider what happens in a film making environment. During the course of film production, the incremental results of work are usually distributed each day to a number of people involved in the movie making activity. Those distributions, known as film dailies, are confidential. If a version leaks out, the studio would like to be able to identify the source of the leak. The source of a leak can be identified by distributing slightly different copies to each recipient, thus uniquely associating each copy with a person receiving it.

This approach can be used in a digital cinema environment where films are distributed to cinemas in digital format rather than celluloid prints. Even though digital distribution of films could be

more flexible, more efficient and less expensive, film producers and distributors are slow to adopt it because of their concern about the potential loss of revenue caused by the illegal copying and redistribution of films. Now, if each cinema receives a uniquely identifiable copy of a film, then, if illegal copies have been made, it should be possible to associate those copies with the offending cinema, and initiate appropriate legal action.

Associating unique information with each distributed copy of digital content is called *fingerprinting*, and watermarking is an appropriate technology to implement it because the embedded watermark which contains this unique information is invisible and inseparable from the content. This type of application is also known as *traitor tracing* because it is useful for monitoring or tracing illegally produced copies of digital work. Since watermarking can be used to keep track of multiple transactions that have taken place in the history of the copy of a digital content, the term *transaction tracking* has been used as well.

10.5 CONTENT AUTHENTICATION

Multimedia editing software makes it easy to alter digital content. Figure 10.3 shows three images. The left one is the original, authentic image. The middle image is a modified version of the original, and the right one shows the image region that has been tampered with. Since it is easy to interfere with a digital content, there is a need to verify the integrity and authenticity of the content.

Figure 10.3. Original image, tampered image, and detection of tampered regions. (Courtesy of D. Kirovski's Power Point presentation).

A solution to this problem can be borrowed from cryptography, where digital signature has been studied as a message authentication method. Digital signature essentially represents some kind of summary of the content. If any part of the content is modified, its summary, the signature, will change making it possible to detect that some kind of tampering has taken place. One example of digital signature technology being used for image authentication is the trustworthy digital camera [33].

Digital signature information needs to be somehow associated and transmitted with a digital content from which it was created. Watermarks can obviously be used to achieve that association by embedding signature directly into the content. Since watermarks used in the content authentication applications have to be designed to become invalid if even slight modifications of digital content take place, they are called *fragile watermarks*. Fragile watermarks, therefore, can be used to confirm authenticity of a digital content. They can also be used in applications where it is important to figure out how digital content was modified or which portion of it has been tampered with. For digital images, this can be done by dividing an image into a number of blocks and creating and embedding a fragile watermark into each individual block.

Digital content may undergo lossy compression transformation, such as JPEG image conversion. Although the resulting JPEG compressed image still has authentic content, the image authenticity test based on the fragile watermark described earlier will fail. *Semi-fragile watermarks* can be used instead. They are designed to survive standard transformations, such as lossy compression, but they will become invalid if a major change, such as the one in Figure 10.3, takes place.

10.6 BROADCAST MONITORING

Many valuable products are regularly broadcast over the television network: news, movies, sports events, and advertisements. Broadcast time is very expensive, and advertisers may pay hundreds of thousands of dollars for each run of their short commercial during the commercial breaks of important movies, series or sporting events. The ability to bill accurately in this environment is very important. It is important to advertisers to ensure that they will pay only for the commercials that were actually broadcast. Also, it is important for

the performers in those commercials, who would like to collect accurate royalty payments from advertisers.

Broadcast monitoring is usually used to collect information about the content being broadcast, and this information is then used as the basis for billing as well as other purposes. A simple way to monitor is to have human observers watch the broadcast and keep track of everything they view. This kind of broadcast monitoring is expensive and prone to errors. Automated monitoring is clearly better. There are two categories of automated monitoring systems, passive and active.

Passive monitoring systems monitor the content being broadcast and attempt to recognize it by comparing it with known content stored in a database. Passive monitoring systems are difficult to implement for a couple of reasons. It is difficult to compare broadcast signals against the database content, and it is expensive to maintain and manage a large database of content for comparison.

Active monitoring systems rely on additional information, broadcast with the content itself, which actually identifies the content. For analog television broadcast, this content identification information can be encoded in the vertical blanking interval (VBI) of the video signal. The problem with this approach is that it is suitable for analog transmission only. It may also not be reliable because, in the USA, content distributors do not have to distribute information embedded in the VBI.

A more appropriate solution for active monitoring is based on watermarking. The watermark containing broadcast identification information gets embedded into the content itself, and the resulting broadcast monitoring solution becomes compatible with broadcast equipment for both digital and analog transmission.

10.7 SYSTEM ENHANCEMENT

Digital watermarking can also be used to convey side-channel information with the purpose of enhancing functionality of the system or adding value to the content in which it is embedded. This type of application, where a device is designed to react to a watermark for the benefit of the user, is also referred to as a device control application [16].

An example of an early application of watermarking for system enhancement is described in the Ray Dolby patent application filed in 1981, where he proposed to make radio devices which would turn the Dolby FM noise reduction control system on and off automatically, in response to an inaudible signal broadcast within the audio frequency spectrum. Such a signal constitutes a simple watermark, and the proposed radio device was an enhancement compared to the radio devices used at that time, where listeners had to manually turn their radio's Dolby FM decoder on and off.

More recently, Philips and Microsoft have demonstrated an audio watermarking system for music. Basically, as music is played, a microphone on a PDA can capture and digitize the signal, extract the embedded watermark and based on information encoded in it, identify the song. If a PDA is network connected, the system can link to a database and provide some additional information about the song, including information about how to purchase it.

Another example of a similar application is Digimarc's MediaBridge system. On the content production side, watermarks representing unique identifiers are embedded into images, and then printed and distributed in magazines as advertisements. On the user side, an image from a magazine is scanned, the watermark is extracted using the MediaBridge software, and the unique identifier is used to direct a Web browser to an associated Web site.

Chapter 11

DIGITAL WATERMARKING CONCEPTS

11.1 DIGITAL WATERMARKING SYSTEMS

A digital watermarking system consists of two main components: the *watermark embedder* and the *watermark detector*, as illustrated in Figure 11.1. The embedder combines the *cover work* C_O, an original copy of digital media (image, audio, video), and the *payload* P, a collection of bits representing the metadata to be added, creating the *watermarked cover* C_W. The watermarked cover C_W is perceptually identical to the original C_O but with the payload embedded inside. The difference between C_W and C_O is referred to as *embedding distortion*. The payload P is not directly added to the original cover C_O. Instead, it is first encoded as a *watermark* W, possibly using a secret key K. The watermark is then modulated and/or scaled, yielding a *modulated watermark* W_m, to ensure that the embedding distortion will be small enough to be imperceptible.

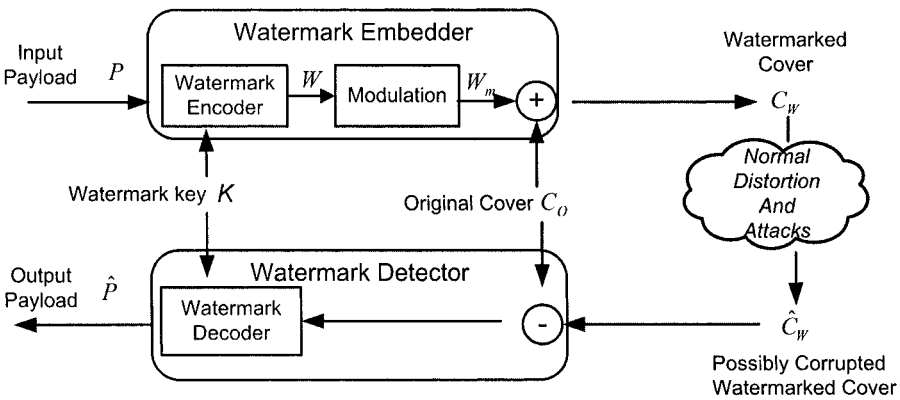


Figure 11.1. Digital Watermarking Systems with Informed Detection.

Before it gets to a detector, the watermarked cover C_W may be subjected to different types of processing, yielding a *corrupted*

watermarked cover $\hat{C}_W$. This corruption can be caused by various distortions from normal signal transformations, such as compression, decompression, D/A and A/D conversions, or by distortions introduced by various malicious attacks. The difference between $\hat{C}_W$ and C_W is referred to as *noise N*.

The watermark detector either extracts the payload $\hat{P}$ from the corrupted watermarked cover $\hat{C}_W$, or it produces some kind of confidence measure indicating how likely it is for a given payload P to be present in $\hat{C}_W$. The extraction of the payload is done with help of a watermark key K.

Watermark detectors can be classified into two categories, *informed* and *blind*, depending on whether the original cover work C_0 needs to be available to the watermark detection process or not. An *informed detector*, also known as a *non-blind detector*, uses the original cover work C_0 in a detection process. A *blind detector*, also known as an *oblivious detector*, does not require knowledge of the original cover C_0 to detect a payload.

11.2 WATERMARKING AS COMMUNICATION

The watermarking system, as presented in the previous section, can be viewed as a form of communication. The payload message P, encoded as a watermark W, is modulated and transmitted across a communication channel to the watermark detector. In this model, the cover work C_0 represents a communication channel and, therefore, it can be viewed as one source of noise. The other source of noise is distortion caused by normal signal processing and attacks.

Modeling watermarking as communication is important because it makes it possible to apply various communication system techniques, such as modulation, error correction coding, spread spectrum communication, matched filtering, and communication with side information, to watermarking. Those communication techniques can be used to help design the key building blocks of a watermarking system which deal with the following design issues:

- How to embed and detect one bit?

- What processing/embedding domain to use?
- How to use side information to ensure imperceptibility?
- How to use modulation and multiplexing techniques to embed multiple bits?
- How to enhance robustness, defined as watermark's resistance to normal signal processing?
- How to enhance security, defined as watermark's resistance to intentional attacks?

11.3 EMBEDDING ONE BIT IN SPATIAL DOMAIN

It is a common practice in communication to model channel noise as a random variable whose values are drawn independently from a Normal distribution with zero mean and some variance, σ_n^2. This type of noise is referred to as Additive White Gaussian Noise (AWGN). It is also known from communication theory that the optimal method for detecting signals in the presence of AWGN is matched filtering, which is based on computing linear correlation between transmitted and received signals and comparing it to a threshold.

Applying those two concepts, the AWGN and matched filtering, to watermarking yields a simple, spatial domain image watermarking technique with blind detection, which is illustrated in Figure 11.2. The watermark is created as an image having the same dimensions as the original cover image C_0 with luminance values of its pixels generated as a key-based pseudo-random noise pattern drawn from a zero mean, unit variance Normal distribution, *N(0,1)*. The watermark is then multiplied with the embedding strength factor *s*, and added to the luminance values of the cover image pixels. The embedding strength factor is used to impose a power constraint in order to ensure that, once embedded, the watermark will not be perceptible. Note that, once the embedding strength factor *s* is selected, it is applied globally to all cover images that need to be watermarked. Also note that this embedding procedure creates the watermark *W* independently of the cover image C_0.

According to the model of the Digital Watermarking System depicted in Figure 11.1, the watermark detector will work on a received image C, which can be represented either as $C = \hat{C}_W = C_O + W_m + N$, if the image was watermarked, or as $C = C_O + N$ otherwise, where N is a noise caused by normal signal processing and attacks.

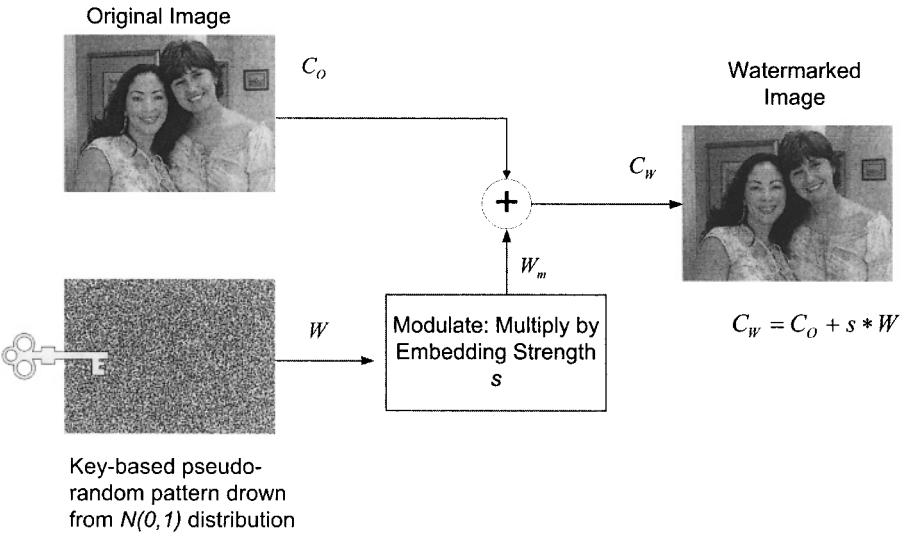


Figure 11.2. Watermark embedding procedure.

To detect the watermark, a detector has to perceive the presence of the signal W in the received, possibly watermarked image C. In other words, the detector has to detect the signal W in the presence of noise caused by C_O and N. Assuming that both C_O and N are AWGN, the optimal method for detecting watermark W in the received image C is based on computing the linear correlation between W and C, as:

$$LC(W, C) = \frac{1}{I \cdot J} W \cdot C = \frac{1}{I \cdot J} \sum_{i,j} w_{ij} c_{ij}, \quad (1)$$

where w_{ij} and c_{ij} represent pixel values at location i,j in W and C, and I and J represent the image dimensions.

If the received image C was watermarked (i.e. if $C = C_O + W_m + N$), then:

$$LC(W,C) = \frac{1}{I \cdot J}(W \cdot C_O + W \cdot W_m + W \cdot N) \quad (2)$$

Because C_O and N are assumed to be AWGN, and the watermark W is created as AWGN, the additive components of linear correlation $W \cdot C_O$, and $W \cdot N$ are expected to have small magnitudes, and the component $W \cdot W_m = sW \cdot W$ is expected to have a much larger magnitude. This is illustrated in Figure 11.3, where it is shown that the AWGNs generated as pseudo-random patterns using different keys (i.e. seeds) have a very low correlation with each other, but a high correlation with itself.

Therefore, if a calculated linear correlation $LC(W,C)$ between the received image C and watermark W is small, then a conclusion can be made that the image C was not watermarked. Otherwise, the image C was watermarked. This decision is usually made based on a threshold T, so that if $LC(W,C) < T$, the watermark W is not detected in C, and if $LC(W,C) \geq T$, the watermark W is detected in C.

A watermark detection procedure based on threshold is illustrated in Figure 11.4. The two curves represent the distribution of linear correlation (*LC*) values calculated for the set of unmarked images (the curve that peaks for the detection value 0), and for the set of watermarked images (the curve that peaks for the detection value 1). For a selected threshold value T, the portion of the curve for the unmarked images to the right of the threshold line T, represents all tested unmarked images which will be erroneously detected as marked images, and the portion of the curve for the marked images to the left of the threshold line T, represents watermarked images which will erroneously be declared as unmarked. The former error is called a *false positive error*, and the latter is called a *false negative error*. The false negative error rate can be considered a measure of efficiency of the watermarking system because it can be seen as a failure rate of the embedder to embed a detectable watermark.

False positive and false negative errors occur because original cover images C_O are not accurately modeled as AWGN, and consequently, they can have a high correlation with the watermark signal W. Several proposed watermarking systems are based on this technique [5][32][60][73][71].

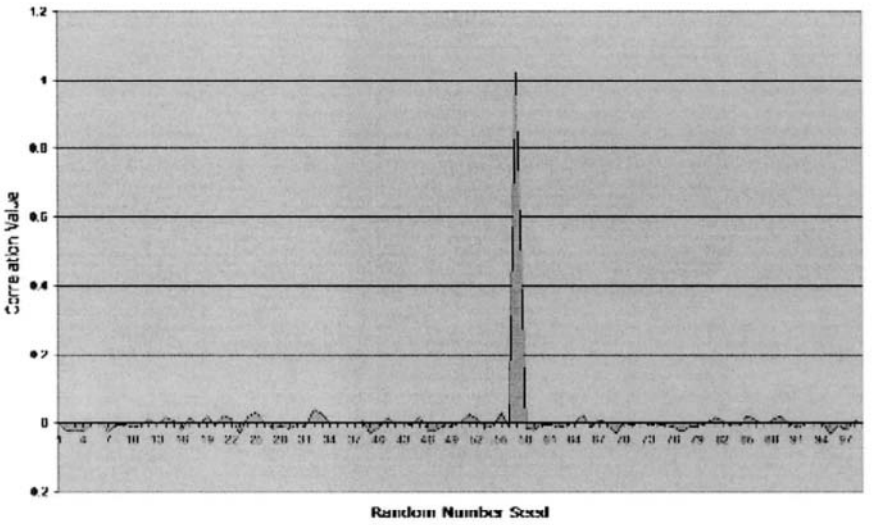


Figure 11.3. Correlation values for a pseudo-random pattern generated with seed=57 correlated with pseudo-random patterns generated with other seeds

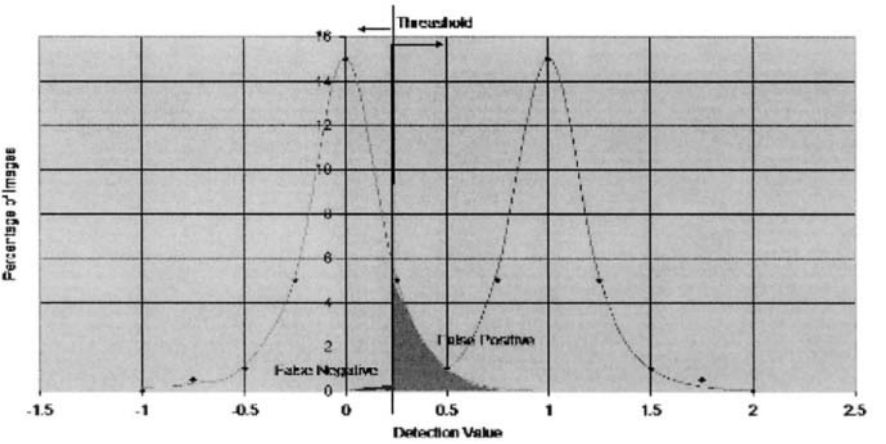


Figure 11.4. Watermark detection procedure based on Linear Correlation (LC).

11.4 PATCHWORK WATERMARKING TECHNIQUE

Patchwork is another spatial domain watermarking technique designed to imperceptibly embed a single bit of information in a cover image [5]. It is a statistical method, which embeds a watermark by changing the statistical distribution of luminance values in the set of pseudo-randomly selected pairs of image pixels. This technique is based on an assumption that luminance values of an image have the following statistical property: $\sum_{n}(a_i - b_i) \approx 0$, where $A = \{a_i\}_1^n$, and $B = \{b_i\}_1^n$ are two patches of pseudo-randomly selected image pixels. A watermark is embedded by increasing the brightness of pixels that belong to the patch $A = \{a_i\}_1^n$, and accordingly, decreasing the brightness of pixels that belong to the patch $B = \{b_i\}_1^n$. In other words, after pseudo-randomly selecting patches $A = \{a_i\}_1^n$ and $B = \{b_i\}_1^n$, luminance values of the selected pixels are modified according to the following formula: $\tilde{a}_i = a_i + \delta \ \wedge \ \tilde{b}_i = b_i - \delta$. This modification creates a unique statistic that indicates presence or absence of watermark. The watermark detector will select the same n pairs of pixels belonging to two patches, and it will compute: $\Delta = \sum_{n}(\tilde{a}_i - \tilde{b}_i)$. If $\Delta \approx 2n\delta$, the image is watermarked; otherwise, it is not.

This watermarking technique creates a watermark independently of the cover image, and it uses blind detection (i.e. the detector does not require the original cover image in order to determine whether the image has been watermarked or not). The watermark detection is based on linear correlation because the detection process described above is equivalent to correlating the image with a pattern consisting of 1's and -1's, where the pattern contains a 1 for each pixel from the patch $A = \{a_i\}_1^n$ and a -1 for each pixel from the patch $B = \{b_i\}_1^n$.

11.5 WATERMARKING IN TRANSFORM DOMAINS

A watermark can be embedded into the cover image in a spatial domain as described in the two techniques above. Alternatively, a watermark embedding operation can be carried out in a transform domain, such as discrete Fourier transform (DFT) domain, the full-

image (global) discrete cosine transform (DCT) domain, the block-based DCT domain, the Fourier-Mellin transform domain, or the discrete wavelet transform (DWT) domain.

Transform domains have been extensively studied in the context of image coding and compression, and many research results can be applied to digital watermarking. The theory of image coding maintains that in most images, the colors of neighboring pixels are highly correlated. Mapping into a specific transform domain, such as DCT or DWT, serves two purposes. It de-correlates the original sample values, and it concentrates the energy of the original signal into just a few coefficients. For example, when a typical image is mapped into the spatial-frequency domain, the energy is concentrated in the low-index terms, which are very large, compared to the high-index terms. This means that a typical image is dominated by the low frequency components. Those low frequencies represent the overall shapes and outlines of features in the image, as well as their luminance and contrast characteristics. High frequencies represent sharp edges and crispiness in the image but contribute little spatial-frequency energy. As an example, a typical image might contain 95% of the energy in the lowest 5% of the spatial frequencies of the two dimensional DCT domain. Retention of these DCT components, together with enough higher frequency components to yield an image with enough sharpness to be acceptable to the human eye, was the objective for the creation of an appropriate quantization table to be used for JPEG compression.

The following sections discuss the selection of a specific transform domain to use for watermarking and the reasons and advantages for each.

11.5.1 DCT Domain and Spread Spectrum Technique

The DCT domain has been used extensively for embedding a watermark for a number of reasons. Using the DCT, an image is divided into frequency bands, and the watermark can be conveniently embedded in the visually important low- to middle-frequency bands. Sensitivities of the human visual system to changes in those bands have been extensively studied in the context of JPEG compression, and the results of those studies can be used to minimize the visual impact of the watermark embedding distortion. Additionally, requirements for robustness to JPEG compression can

be easily addressed because it is possible to anticipate which DCT coefficients will be discarded by the JPEG compression scheme. Finally, since JPEG/MPEG coding is based on DCT decomposition, embedding a watermark in the DCT domain makes it possible to integrate watermarking with image and video coding. This integration enables development of real-time watermarking applications.

An efficient solution for watermarking in the global DCT domain was introduced by Cox et. al. [17], and it is based on spread spectrum technology. The general spread spectrum system spreads a narrow-band signal over a much wider frequency band so that the signal-to-noise ratio (SNR) in a single frequency band is low and appears like noise to an outsider. However, a legitimate receiver with precise knowledge of the spreading function should be able to extract and sum up the transmitted signals so that the SNR of the received signal is strong.

As stated earlier, a watermarking system can be modeled as communication, where the cover image is treated as noise, and the watermark is viewed as a signal that is transmitted through it. This modeling makes it possible to apply techniques that worked in spread spectrum communications, to watermarking. The basic idea is to spread the watermark energy over visually important frequency bands, so that the energy in any one band is small and undetectable, making the watermark imperceptible. Knowing the location and content of the watermark, makes it possible to concentrate those many weak watermark signals into a single output with high watermark-to-noise ratio (WNR). Here is a high-level overview of this watermarking technique.

The watermark is embedded in the first n lowest frequency components $C = \{c_i\}_1^n$ of a full image DCT in order to provide high level of robustness to JPEG compression. The watermark consists of a sequence of real numbers $W = \{w_i\}_1^n$ drawn from a Normal distribution $N(0,1)$, and it is embedded into the image using the formula $\tilde{c}_i = c_i(1 + sw_i)$, where s is the watermark embedding strength factor. Watermark detection is performed using the following similarity measure:

$$sim(W,W') = \frac{W \cdot W'}{\sqrt{W' \cdot W'}}. \quad (3)$$

The W' is the extracted watermark, calculated as:

$$\{w_i'\}_1^n = \left\{ (\frac{\tilde{c}_i}{c_i} - 1)/s \right\}_1^n, \quad (4)$$

where the $\tilde{c}_i$ components are extracted from the received, possibly watermarked image, and the c_i components are extracted from the original cover image. The watermark is said to be present in the received image if $sim(W,W')$ is greater than the given threshold.

Because the original image components c_i are used for calculation of the extracted watermark W', which is used for watermark presence tests, this watermarking system falls into the category of systems with informed detectors.

An empirically selected value of 0.1 was used for the embedding strength factor s, and the watermark was spread across the 1000 lowest-frequency non-DC DCT coefficients (n=1000). Robustness tests showed that watermarks embedded using this watermarking scheme survived JPEG compression to the quality factor of 5%, dithering, fax transmission, printing, photocopying, scanning, multiple watermarking, and collusion attacks.

11.5.2 Wavelet Domain

With the standardization of JPEG-2000, and a decision to use wavelet-based image compression instead of DCT-based compression, watermarking techniques operating in the wavelet transform domain have become more attractive to the watermarking research community. The advantages of embedding watermarks in the wavelet transform domain are an inherent robustness of the embedded mark to the JPEG-2000 lossy compression, and the possibility of minimizing computation time by embedding watermarks inside of a JPEG-2000 encoder. Additionally, the wavelet transform has properties that can be exploited by watermarking solutions. For instance, wavelet transform provides multi-resolution representation of images. This can be exploited to

build more efficient watermark detection schemes, where watermark detection starts from the low-resolution sub-bands first, and only if detection fails in those sub-bands, does it explore the higher resolution sub-bands and the additional coefficients it provides.

Zhu et al. [72] introduces a unified approach to digital watermarking of images and video based on the two-dimensional (2-D) and three-dimensional (3-D) discrete wavelet transforms. This approach is very similar to that of Cox et al. [17] presented earlier. The only difference is that Zhu generates a random vector with *N(0,1)* distribution and spreads it across coefficients of all high-pass bands in the wavelet domain as a multi-resolution digital watermark, whereas Cox does it only across a small number of perceptually most important DCT coefficients. The watermark added to a lower resolution represents a nested version of the watermark corresponding to a higher resolution. The hierarchical organization of the wavelet representation allows detection of watermarks at all resolution levels except the lowest one. The ability to detect lower-resolution watermarks reduces computational complexity of watermarking algorithms because fewer frequency bands are involved in computation. It also makes this watermarking scheme robust to image and video down-sampling operation by a power of 2 in either space or time.

11.5.3 DFT Domain

The discrete Fourier transform of an image is generally complex valued, and this leads to a magnitude and phase representation for the image. Most of the information about any typical image is contained in the phase, and the DFT magnitude coefficients convey very little information about the image [65].

Adding a watermark to the phase of the DFT [66] improves the robustness of the watermark because any modification of those visually important image components in an attempt to remove the watermark will significantly degrade the quality of the image. Another reason to modify or modulate the phase coefficients to add a watermark is based on communications theory research results, which established that modulating the phase is more immune to noise than modulating amplitude. Finally, the phase-based watermarking is relatively robust to changes in image contrast.

An alternative is to add a watermark to the DFT magnitude coefficients only [65]. Adding watermark to DFT magnitude coefficients is beneficial because DFT magnitude coefficients convey very little information about an image, and embedding a watermark in those coefficients should not introduce a perceptible distortion. However, since distortion caused by modifications of the DFT magnitude coefficients is much less perceptible than distortion caused by phase modifications, it is expected that good image compressors would give much higher importance to preserving the DFT phase then the DFT magnitude, rendering the DFT magnitude-based watermarking system vulnerable to image compression. Surprisingly enough, this is not the case, and all major compression schemes, such as JPEG, Set Partitioning in Hierarchical Trees (SPIHT), and MPEG preserved the DFT magnitude coefficients as well as they preserved the DFT phase [65].

The DFT magnitude domain is translation invariant, and this property can be very useful for watermarking. The DFT magnitude domain is translation invariant because a cyclic translation of an image in the spatial domain does not affect the DFT magnitude coefficients. Consequently, the watermark embedded into DFT magnitude coefficients will not be affected by image translations or shifts in spatial domain.

Image translation, as well as image scaling and rotation, generally do not affect a perceived image quality. Nevertheless, translation or any other geometrical transformation desynchronizes the image and thus makes the watermarks embedded using techniques described in the previous subsections undetectable. To make the watermark detectable after a geometrical transformation has been applied to the watermarked image, the watermark needs to be synchronized. The synchronization process consists of an extensive search over a large space that covers all possible x-axis and y-axis translation offsets, all possible angles of rotation, and all possible scaling factors. There is an alternative to searching for synchronization during the watermark detection process [58]. The basic idea is to avoid a need for synchronization search by transforming the image into a new workspace that is invariant to specific geometrical transformations and embedding the watermark in that workspace. This is shown in Figure 11.5. For example, watermarks can be embedded in the Fourier-Mellin transform domain, which is invariant to translation, scaling and rotation. The Fourier-Mellin transform is computed by

taking the Fourier transform of a log-polar map. A log-polar mapping is defined as:

$$u = e^{\mu}\cos(\theta)$$
$$v = e^{\mu}\sin(\theta) \tag{5}$$

It provides one-on-one mapping between $(u,v)\in\Re^2$ and (μ,θ), $\mu\in\Re, \theta\in(0,2\pi)$, so that scaling and rotation in the *(u,v)* space is converted into translation in the (μ,θ) space. The (μ,θ) space is converted into the DFT magnitude domain to achieve translation invariance, and the watermark can be embedded in that domain using one of watermarking techniques described earlier.

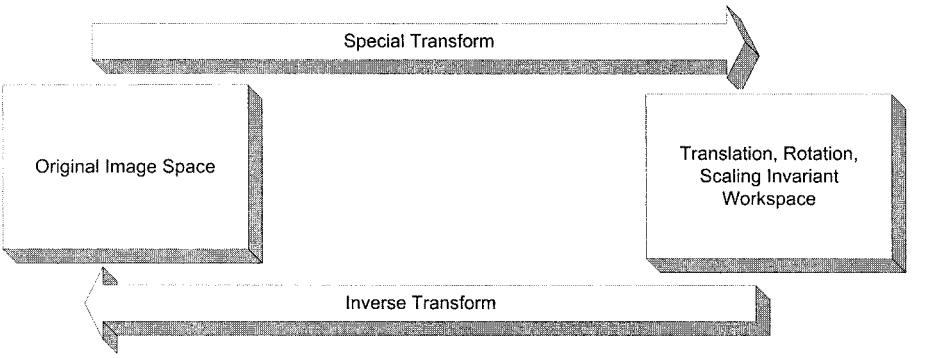


Figure 11.5. Robustness to geometric transformations: Embed the watermark inside a new workspace, which is invariant to translation, rotation and scaling

11.6 SIDE INFORMATION AND INFORMED EMBEDDING

The embedding components of the watermarking systems described so far create a watermark *W* independently of the cover *Co*. The embedder depicted in Figure 11.2, pseudo-randomly generates the watermark pattern first, and then multiplies each watermark pixel value with a global embedding strength factor *s*, and adds it to the cover *Co*. The global embedding strength factor *s* is also selected independently of the cover *Co*, and it is used to control a trade-off between watermark robustness and its transparency or imperceptibility. Increasing the embedding strength factor *s* will

increase the energy of the embedded watermark, resulting in higher robustness. Yet, it will also increase embedding distortion resulting in a less transparent watermark and causing a loss of fidelity of the watermarked image C_W compared to the original cover C_O.

The embedder obviously has access to the original cover image C_O in order to be able to embed the watermark into it. However, even though the embedder has access to the original cover image, the watermarking systems described so far did not take advantage of that information. This section describes how watermarking systems can use the information about the original cover image to improve watermark embedding performance. This kind of watermarking systems is called watermarking systems with *informed embedding*, and the model is presented in Figure 11.6.

This model improves the effectiveness of the watermarking embedder depicted in Figure 11.2 by taking into consideration the original cover image on the embedding side, and calculating the embedding strength factor s according to this information.

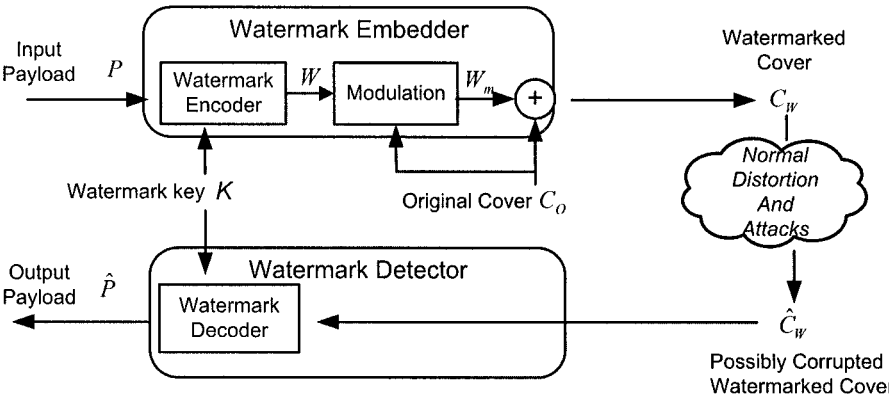


Figure 11.6. Digital Watermarking System with Informed Embedding and Blind Detection.

It was demonstrated earlier that the original watermarking system was not 100% effective, because it had a non-zero false negative rate. This is not surprising, because when a watermark is created independently of the cover image, it is natural that some cover images can interfere with the watermark in such a way that the embedded watermark is not detectable. This can be corrected by

taking into consideration the characteristics of the cover work while creating the watermark. This is called the informed embedding, and it can be used to create a watermarking system that yields 100% effectiveness. A simple modification of the previously described watermarking system can make the scheme 100% effective. The embedding strength s can be adjusted appropriately for each individual cover image, so that every watermarked cover C_W has fixed-magnitude linear correlation with the watermark W. In other words the embedder selects the embedding strength factor s, to ensure that $LC(W, C_W) \geq T$ is always correct.

The design of a watermarking system with informed embedding can be presented as an optimization problem which can be stated as follows: Given an original cover C_O, select the embedding strength s to maximize specific important property, such as fidelity, robustness, or embedding effectiveness, while keeping the other property or properties fixed. For example, informed embedding can be used to improve robustness of the watermark while maintaining a fixed fidelity. The objective is to maximize the energy of the watermark signal, without increasing perceptible distortion of the watermarked signal, by taking advantage of imperfections of human visual system (HVS), and its inability to recognize all the changes equally. The characteristics of HVS and its sensitivities to frequency and luminance changes, as well as its masking capabilities have been captured into various *perceptual models* (i.e. models of HVS). Those models are then used as part of watermark embedding algorithms to help identify areas in the original cover image where the watermark embedding strength factor can be locally increased without introducing a perceptible change.

The HVS response to frequency and luminance changes has been heavily researched. Frequency sensitivity refers to the eye's response to spatial-, spectral-, or time-frequency changes. Spatial frequencies are perceived as patterns or textures, and spatial-frequency sensitivity is usually described as the eye's sensitivity to luminance changes [16]. It has been shown that an eye is the most sensitive to luminance changes in the mid-range spatial frequencies, and that sensitivity decreases at lower and higher spatial frequencies. The pattern orientation affects sensitivity as well. An eye is most sensitive to vertical and horizontal lines and edges, and it is least sensitive to lines and edges with 45° orientation. Spectral frequencies are perceived as colors, and the human eye is least

sensitive to changes in blue color. Hurtung and Kutter [38] took into consideration color sensitivity of HVS, and proposed a solution where the watermark is added to the blue channel of an RGB image. Temporal frequencies are perceived as motion or flicker, and it has been demonstrated that eye sensitivity decreases very quickly as temporal frequencies exceed 30 Hz.

A number of solutions have been proposed in which the frequency sensitivity of the HVS is exploited to ensure that the watermark is imperceptible. Those solutions use transform domain (e.g. DCT, DFT, wavelet), and the watermark is added directly into the transform coefficients of the image. Even when a global embedding strength factor s, is used by the embedder, the watermark embedding algorithm can be changed to take into account local characteristics of the cover C_O as follows. If $W = \{w_i\}$ is the watermark, $C_O = \{c_i\}$ is the cover image, $C_W = \{\tilde{c}_i\}$ is the watermarked image, and s represents the global embedding strength, then the embedder can embed the watermark using the following formula: $\tilde{c}_i = c_i(1 + sw_i)$. Here, the amount of change is clearly dependent on characteristics of the cover image C_O. Contrast that with the embedding formula $\tilde{c}_i = c_i + sw_i$ presented earlier, where the amount of change was the same, irrespective of the magnitude of the c_i coefficients.

More advanced embedding algorithms have been created by taking full advantage of characteristics of the HVS. It is known that different spectral components may have different levels of tolerance to modification, and also it is known that the presence of one signal can hide or mask the presence of another signal. Those characteristics of the HVS can be exploited as well to create an efficient image-adaptive solution. A single embedding strength factor, s, is not appropriate in that case. Instead, the more general watermark embedding formula $\tilde{c}_i = c_i(1 + s_i w_i)$ should be used. Different image-adaptive solutions select multiple scaling parameters s_i in different ways. More information about various image-adaptive watermarking solutions can be found in [74].

11.7 SIDE INFORMATION AND INFORMED CODING

Informed embedding watermarking systems use the cover information as part of the watermark embedding operation. The watermark W is created independently of the cover C_0, and then it is locally amplified or attenuated depending on the local characteristics of the cover C_0 and based on perceptual models of sensitivities and masking capabilities of the HVS. The watermark W is still created independently of the cover C_0, and because of that it is clear that those algorithms do not take full advantage of all the side information about the cover C_0 available to the watermark embedder.

Instead of creating the watermark independently of the cover C_0, and then modifying it based on local characteristics of the cover C_0 to minimize interference and distortion, the embedder can use the side information about the cover C_0 during the watermark creation and encoding process to choose between several alternative watermarks and select the one that will cause the least distortion of the cover C_0. This technique is referred to as watermarking with *informed coding*.

Watermarking with informed coding was inspired by theoretical results published by Max Costa in his "Writing on Dirty Paper" report [15] on the capacity of a Gaussian channel having interference that is known to the transmitter. Costa described the problem using a dirty paper analogy, which can be stated as follows [16]: Given a sheet of paper covered with independent dirt spots having normally distributed intensity, write a message on it using a limited amount of ink, and then send the paper on its way to a recipient. Along the way the paper acquires more normally distributed dirt. How much information can be reliably sent, assuming that the recipient cannot distinguish between ink and dirt? This problem is illustrated in the Figure 11.7.

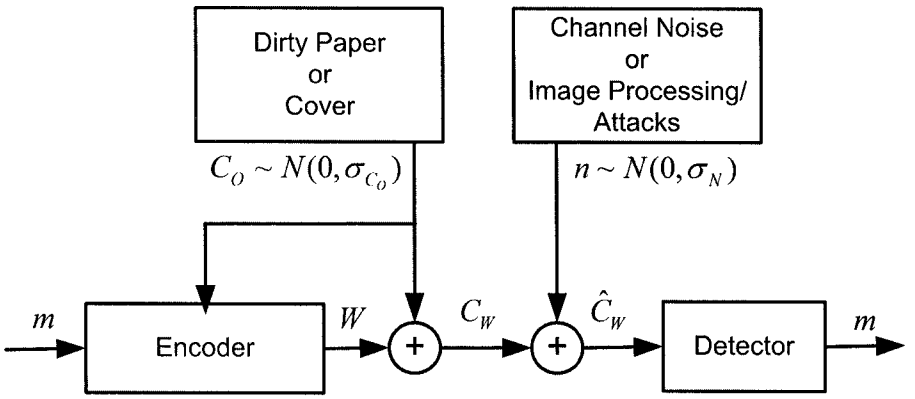


Figure 11.7. Dirty-paper channel studied by Costa. There are two noise sources, both AWGN. The encoder knows the characteristics of the first noise source (dirty paper or the original cover) before it selects (watermark) *W*

Costa showed that the capacity of his dirty-paper channel is given by:

$$C = \frac{1}{2}\log_2\left(1+\frac{P}{\sigma_N^2}\right) \tag{6}$$

where *P* represents the power constraint imposed on the transmitter (i.e. there was a limited amount of ink available to write a message), and σ_N^2 represents a variance of the second source of noise. Surprisingly enough, the first source of noise, the original dirt on the paper, does not have any effect on the channel capacity.

Costa's dirty paper problem can also be viewed as a watermarking system with blind detection. The message to be written is a watermark *W*, the message is written on (embedded in) the first coat of dirt, the cover C_O, the limited amount of ink can be interpreted as a power constraint that ensures fidelity, and the second noise source, *n*, represents distortions caused by normal signal processing and attacks. Since the dirty paper channel can be cast as a watermarking system, Costa's results attracted much attention in the watermarking research community because his results established an upper bound for watermark capacity and demonstrated that capacity does not depend on the interference caused by the cover C_O.

Watermarking systems inspired by Costa's work are based on the following principle: Instead of having one watermark for each message, have several alternatives available (the more, the better), and select the one with minimum interference with the cover C_0. Unfortunately, straightforward implementation of this principle is not practical. The problem is that both the watermark embedder and the watermark detector are required to find the closest watermark to a given vector representing possibly distorted cover C_0, for every message or payload. For a large number of messages and a large number of watermarks for each message, computational time and storage requirements are simply too high.

This problem can be solved by arranging watermarks in such a way as to allow efficient search for the closest watermark to a given cover C_0 [29]. A lattice has been identified as an appropriate tool for structuring watermarks, and most of the work related to the watermarking with informed coding is based on the use of lattice codes, where watermarks represent points in a regular lattice.

Chen and Wornell [13] proposed watermark embedding based on that principle. Their method called *quantization index modulation* (QIM) is based on the set of N-dimensional quantizers, one quantizer for each possible message m that needs to be transmitted. The message to be transmitted determines the quantizer to use. The selected quantizer is then used to embed the information by quantizing the cover C_0. The quantization of C_0 can be done in any domain (i.e. spatial, DCT, etc.). A distortion can be controlled by selecting an N-dimensional quantization point closest to the cover C_0. In the decoding process, a distance metric is evaluated for all quantizers and the one with the smallest distance from the received image $\tilde{C}_W$ identifies the embedded information. This is illustrated in the Figure 11.8, for a one-dimensional case and two uniform, scalar quantizers representing two different messages, *m1* and *m2*. Those two messages can be used to represent two distinct values of one bit: 0 and 1. The watermarking system based on QIM was shown to have better performance than other watermarking systems based on the standard spread-spectrum modulation, which is not image-adaptive.

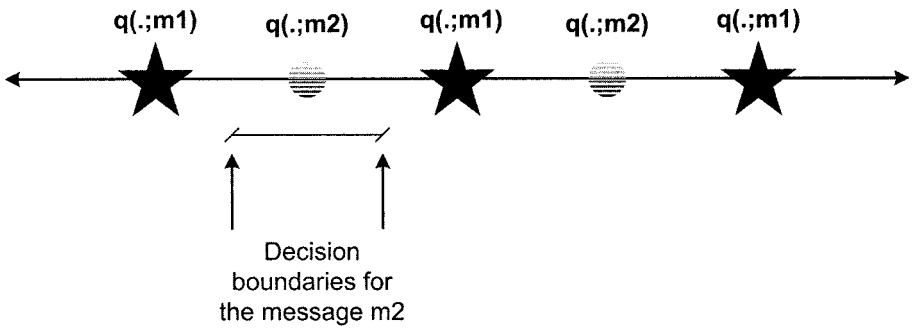


Figure 11.8. Quantization index modulation information embedding and detection.

There are other possible ways to partition the space and to embed a watermark. For instance, it is possible to embed a watermark by enforcing a desired relationship [44]. This solution uses the relationship between DCT coefficients to embed a watermark as follows: An image is partitioned into 8×8 blocks, and out of set of DCT coefficients $(a_{11},..,a_{88})$, the pair (a_{ij},a_{mn}) is selected to represent a single bit in each block. The mutual relationship between two coefficients can then be used to represent the value of one bit. For example, the $a_{ij} < a_{mn}$ relationship can be selected to represent bit '1'. The bit embedding process then consists of making appropriate changes to the pair of coefficients, if needed, to ensure that the relationship between coefficients is correct for the bit value that needs to be embedded. Assuming that the relationship $a_{ij} < a_{mn}$ represents bit value '1', the embedding algorithm can be described as follows: To embed a bit value '1' into the 8×8 block, check the relationship between coefficients in the pair. If $a_{ij} < a_{mn}$, nothing needs to be done since the relationship already indicates a correct bit value. Otherwise, modify coefficients appropriately to enforce desired relationship $a_{ij} < a_{mn}$. In order to strike the balance between robustness and possible image degradation caused by modifications of the coefficients, the pair is selected from mid-range frequencies. This approach shows good robustness to JPEG compression down to a quality factor of 50%.

11.8 WATERMARKING AND MULTI-BIT PAYLOAD

A watermark designed to carry only one bit of information is typically created as a pseudo-random noise drawn from Gaussian distribution, as described earlier. The detector extracts the embedded bit by verifying whether the watermark is present or not. Most watermarking applications, however, require more than one bit of information to be embedded.

Creating multiple watermarks and associating individual watermarks to different bit strings can increase the information rate of the watermarking system. To support 4-bit messages, 16 (= 2^4) different watermarks need to be created, so that each one can be associated with one of the 16 possible 4-bit messages. The message is detected by computing a detection value for each of 16 watermarks and then selecting the one with the highest detection value. This technique is known as *direct message coding*, and it works well for short messages. However, it is not practical for longer bit strings. For example, in order to embed 16 bits of information, the watermarking system would need 2^{16} = 65536 different watermarks. Those watermarks would have to be created with the maximum possible separation to avoid a situation where a small corruption in a watermarked image leads to the detection of an incorrect watermark. It is not easy to ensure that 65536 watermarks are far apart from one another. Additionally, watermark detection is not simple, because the detector needs to compare a test image against 65536 different watermarks even if it only has to check for the watermark absence.

An alternative to direct message coding is a technique in which a different watermark represents each individual bit of the multi-bit message. A multi-bit message is embedded into a cover image by adding watermarks representing individual bits of the multi-bit message one by one [44]. More generally, this technique can be viewed as the scheme where watermarks representing individual bits of a multi-bit message are first combined together into a single watermark representing the whole message, and then added into the cover image.

Watermarks can be combined together in different ways. They can be tiled together so that any individual tile is a watermark representing individual message bit. This watermarking scheme is

equivalent to space division multiplexing. Alternatively, an approach equivalent to frequency division multiplexing can be used by placing watermarks representing individual message bits into disjoint frequency bands. Or, most generally, an approach analogous to code division multiplexing in spread spectrum communications can be used. In this case, a watermark representing an individual message bit is spread across the whole image. Multiple watermarks representing individual message bits can be combined together without interfering with each other because they are selected to be mutually orthogonal.

11.9 CLASSIFICATION OF WATERMARKING SYSTEMS

Watermarking systems can be classified according to many different criteria. Depending on whether a watermark embedder uses side information or not, watermarking systems can be categorized into systems with *blind* or *informed* embedders. The informed embedder category can further be divided into embedders with *informed embedding* and embedders with *informed coding*. Embedders with informed embedding generate a watermark independently of the cover, but use the information about the cover to optimize the watermark embedding operation. Embedders with informed coding select the most appropriate watermark for the given cover work. Watermarking systems can also be categorized into systems using *informed* or *blind* detectors, depending on whether the original cover work is needed to detect the watermark. Other classifications are possible as well. Watermarking systems can be classified based on how a watermark is merged with the cover work to create the watermarked cover, what technology is used to minimize perceptible distortion of the watermarked cover, whether watermarks are manipulated in spatial or transform domains, or based on how they implement support for multi-bit messages. The classification summary is shown in the Table 11.1.

Table 11.1. Classification of Watermarking Systems

Criteria	Categories	Characteristics
Watermark Embedding	Blind	Watermark is selected independently of the of the Cover Work, and it is embedded independently of the Cover Work.
	Informed Embedding	Watermark is selected independently of the Cover Work, but the Cover Work information is used to optimize the watermark embedding operation.
	Informed Coding	Watermark is selected based on the Cover Information. This is done by having a number of different watermarks available for embedding, and selecting the closest one to the given Cover Work. The closest watermark is the one with the minimum interference with the given Cover Work.
Watermark and Cover Merging	Addition	Watermark signal is simply added to the Cover Work. This addition can either be Blind or Informed embedding. Watermark signal can be added to the luminance channel or to the color channels, and this addition can take place in different workspace domains.
	Quantization	Informed Coding that uses lattice codes to allow an efficient search for the closest Watermark to a given Cover Work. The watermark candidates represent points in a regular lattice, and the Cover Work is quantized to that lattice.
	Masking	Informed Embedding that takes advantage of the properties of the Human Visual System to optimize the watermark embedding operation. Optimization can maximize watermark energy while keeping a visual distortion of the Watermarked Cover constant, or alternatively, it can minimize visual distortion while keeping the watermark energy constant.
Main Technologies	Spread Spectrum	Addition-based Watermarking method that uses spread spectrum technology to maximize security and robustness of the embedded watermark and minimize distortion of the Watermarked Cover. The watermark energy is spread across visually

			important frequency bands, so that the energy in any one band is small and undetectable, making the embedded watermark imperceptible. However, knowing the location and the content of the watermark makes it possible to concentrate those many weak watermark signals into a single signal with high watermark to noise ration.
	Quantization Index Modulation		Quantization-based Watermarking method that uses a set of N-dimensional quantizers, one quantizer for each possible message *m* (i.e. watermark *W*) that needs to be transmitted.
Watermark Detection	Blind or Oblivious		Watermarking System that does not require the original Cover Work to be able to detect the embedded watermark.
	Informed		Watermarking System that uses the original Cover Work in the watermark detection process.
Workspace Domain	Spatial		Watermark embedding takes place in the spatial domain.
	Transform	DCT	Watermark embedding takes place either in a global or block DCT domain. In a DCT domain, the Cover Work is divided into frequency bands, and the watermark can be embedded either into low, medium or high frequencies, depending on how robust and perceptible the embedded watermark is required to be. Additionally, watermarks embedded in the DCT domain are inherently robust to the JPEG lossy compression
		Wavelet	Watermark embedding takes place in the Wavelet transform domain, which provides multi-resolution representation of the Cover Work. Embedding watermarks hierarchically, starting from the low-resolution sub-bands first, makes it possible to implement successive decoding of the watermark, where higher-resolution sub-bands are consulted only if the watermark was not detected in the lower-resolution sub-bands. Additionally, watermarks embedded in the wavelet transform domain are inherently robust to the JPEG-2000 lossy compression.

		DFT	Watermark embedding takes place in the DFT domain, where the watermark signal is added either to the phase coefficients or magnitude coefficients of the DFT. The phase coefficients have been used for robustness, because the phase contains most of the energy of the typical image. The magnitude coefficients have been used because they are not affected by image cyclic translation.
Multi-Bit Watermarking	Direct Message Coding		Each multi-bit message is mapped to an individual, uniquely detectable watermark.
	Bit Coding	Space Division	Individual message bits are mapped into watermarks. The Cover Work is divided in space into the equal sized blocks, and watermarks representing individual message bits are embedded into different blocks, one watermark per block.
		Frequency Division	Individual message bits are mapped into watermarks, and watermarks representing those individual message bits are placed into disjoint frequency bands.
		Code Division	Individual message bits are mapped into watermarks, and watermarks representing those individual message bits are spread across the whole Cover Work. The embedded watermarks will not interfere with each other because they have been selected to be mutually orthogonal

11.10 EVALUATION OF WATERMARKING SYSTEMS

Once a watermarking system has been designed and implemented, it is important to be able to objectively evaluate its performance. This evaluation should permit the comparison of results against other watermarking systems designed for the same or similar purpose [40][43][59].

By definition, watermarking is a technique for embedding a watermark into a cover work imperceptibly and robustly. Therefore, the quality of a new watermarking system can be measured by evaluating those two properties and then comparing the results against an equivalent set of measures obtained by evaluating other

watermarking systems. But, how does one objectively measure whether a distortion introduced by embedding a watermark is perceptible or not? This is not easy. Watermark imperceptibility can be evaluated either using subjective evaluation techniques involving human observers, or using some kind of distortion or distance metrics. The former cannot be automated and the later is not always dependable. Watermark robustness is easier to evaluate thanks to the existence of standardized benchmark tests. Those tests are designed to create various distortions to the watermarked cover under tests, so that it is possible to measure watermark detection rate under those conditions.

In addition to imperceptibility and robustness, watermarks have other properties that may need to be evaluated.

11.10.1 Evaluation of Imperceptibility

Imperceptibility of an embedded watermark can be expressed either as a measure of *fidelity* or *quality*. Fidelity represents a measure of similarity between the original and watermarked cover, whereas quality represents an independent measure of acceptability of the watermarked cover. The most accurate tests of fidelity and quality are subjective tests involving human observers. Those tests have been developed by psychophysics, a scientific discipline whose goal is to determine the relationship between the physical world and people's subjective experience of that world. An accepted measure for evaluation of the level of distortion is a *Just Noticeable Difference* (JND), and it represents a level of distortion that can be perceived in 50% of experimental trials. Thus, one JND represents a minimum distortion that is generally perceptible.

Watermark perceptibility can be measured using different experiments developed as a result of various psychophysics studies. One such experiment is called the *two alternative, forced choice test.* In this experiment, human observers are presented with a pair of images, one original and one watermarked, and they must decide which one has higher quality. The responses are statistically analyzed and results of that analysis provide some information about whether the embedded watermark is perceptible. If the fidelity of the watermarked image is high, meaning that it is very similar to the original image, responses will be random and approximately 50% of the observers will select the original image as the higher-quality

one, and 50% of the observers will select the watermarked image as the higher-quality image. This result can be interpreted as zero JND. If the watermark strength is increased, the perceptible distortion will increase as well. With that, the ratio of observers identifying the original image as the higher quality one will increase as well. Once this ratio gets to 75%, the distortion is equivalent to one JND. Variations of that test are possible, and more information about it can be found in [16].

Another, more general approach allows observers more options in their choice of answers. Instead of selecting the higher-quality image, observers are asked to rate the quality of the watermarked image under test. One example of quality scale that can be used to evaluate perceptibility of an embedded watermark is the scale recommended by the ITU-R Rec. 500, in which a quality rating depends on the level of impairment created by distortion. The recommended scale has 5 quality levels, which go from excellent to bad, and those quality levels correspond to impairment descriptions, which go from imperceptible distortion to very annoying distortion.

These subjective tests can provide a very accurate measure of perceptibility of an embedded watermark, but they can be expensive to administer, they are not easily repeatable, and they cannot be automated.

An alternative approach is an automated technique for quality measure based on a model of the HVS. One such model was proposed by Watson [70]. Watson's model estimates the perceptibility of changes in terms of changes of individual DCT blocks, and then it pools those estimates into a single estimate of perceptual distance $D(C_O, C_W)$, where C_O is the original image, and C_W is a distorted version of C_O.

The model has three components: sensitivity table, luminance masking and contrast masking. The sensitivity table, derived in [1], specifies the amount of change for each individual DCT coefficient that produces one JND. But it is known that sensitivity to coefficient change depends on the luminance value, so that in bright background DCT coefficients can be changed by a larger amount before producing one JND. In other words, the bright background can mask more noise than the dark background. To account for this,

Watson's model adjusts the sensitivity table S_{ij} for each block k, according to the block's DC term, as follows:

$$SL(i,j,k) = S(i,j) \cdot \left[\frac{C_o(0,0,k)}{\overline{C}_o} \right]^{\alpha}, \tag{7}$$

where $C_o(0,0,k)$ is the DC values of the k^{th} block, α is a constant with a suggested value of 0.649, and $\overline{C}_o$ is the average of the DC coefficients in the image.

The third component of the model, the contrast masking, represents the reduction in visibility of change in one frequency due to the energy present in that frequency. The contrast masking is accounted for as follows:

$$SLC(i,j,k) = \max\{SL(i,j,k), |C_o(i,j,k)|^{v(i,j)} SL(i,j,k)^{1-v(i,j)}\} \tag{8}$$

where $v(i,j)$ is a constant between 0 and 1 and may be different for each frequency coefficient. Watson uses a value of 0.7 for all i,j. The $SLC(i,j,k)$ represents the amounts by which individual terms of the block DCT may be changed before resulting in one JND.

To compare the original image C_o and a distorted image C_w, the model first computes the difference between corresponding DCT coefficients,

$$e(i,j,k) = C_w(i,j,k) - C_o(i,j,k), \tag{9}$$

and then uses it to calculate the error in the i,j^{th} frequency of the block k as a fraction of one JND given by:

$$d(i,j,k) = \frac{e(i,j,k)}{SLC(i,j,k)} \tag{10}$$

Those individual errors are then combined, or pooled together, into a single perceptual distance measure:

$$D(C_O, C_W) = \left(\sum_{i,j,k} |d(i,j,k)|^p \right)^{1/p}, \tag{11}$$

where Watson recommends a value of $p = 4$.

In general, modeling of HVS is very complex and so far, the resulting quality metrics did not show a clear advantage over simple distortion metrics [69].

Distortion metrics is yet another alternative. It is based on measuring distortion caused by embedding a watermark, and it is very easy to apply. The distortion can be represented as a measure of difference or distance between the original and the watermarked signal. One of the simplest distortion measures is the mean squared error (MSE) function defined as:

$$MSE(C_W, C_O) = \frac{1}{N} \sum_N (c_w[i] - c_o[i])^2 \cdot \tag{12}$$

The most popular distortion measures are the signal-to-noise ratio (SNR) defined as:

$$SNR(C_W, C_O) = \sum_N c_o^2[i] / \sum_N (c_O[i] - c_W[i])^2, \tag{13}$$

and the peak SNR defined as:

$$PSNR(C_O, C_W) = \max_N c_o^2[i] / \sum_N (c_o[i] - c_W[i])^2 \cdot \tag{14}$$

A more detailed list of distortion measures is presented in [40].

Distortion metric tests are simple and popular. Their advantage is that they do not depend on subjective evaluations. Their disadvantage is that they are not correlated with human vision. In other words, a small distance between the original and the watermarked signal does not always guaranty high fidelity of the watermarked signal.

Wang and Bovik [69] have proposed a new quality metric called the *Universal Image Quality Index (UIQI).* This quality index is calculated

by modeling any image distortion as a combination of the following three factors: loss of correlation, luminance distortion and contrast distortion. It is defined as:

$$Q = \frac{4\sigma_{xy}\bar{x}\bar{y}}{(\sigma_x^2 + \sigma_y^2)(\bar{x}^2 + \bar{y}^2)}, \tag{15}$$

where x is the original image, y is a distorted version of x, and

$$\bar{x} = \frac{1}{N}\sum x_i, \quad \bar{y} = \frac{1}{N}\sum y_i, \quad \sigma_x^2 = \frac{1}{N-1}\sum (x_i - \bar{x})^2, \quad \sigma_y^2 = \frac{1}{N-1}\sum (y_i - \bar{y})^2$$

Even though the UIQI is mathematically defined, and it is not explicitly based on the HVS model, it performs significantly better than the widely used MSE distortion metric.

11.10.2 Evaluation of Other Properties

The robustness property of a watermark can be evaluated by applying various kinds of "normal" signal distortions and attacks that are relevant for the target application. Robustness can be assessed by measuring the detection probability of the watermark after signal distortion. This is usually done using standardized benchmarking tests.

Reliability can be evaluated by assessing the watermark detection error rate. This can be done either analytically, by creating models of watermarking systems under test, or empirically, by running a number of tests and counting the number of errors. A watermark detector can make two types of detection errors, *false positive* and *false negative*. False positive errors occur when the watermark detector incorrectly indicates that a watermark is present, and false negative errors occur when the detector fails to identify an existing watermark. False positive and false negative errors are interrelated, and it is not possible to minimize the occurrence rates of both error types simultaneously. Because of that, both error types should always be measured and presented together. A *receiver operating characteristics* (ROC) curve is one possible presentation.

The capacity of a watermarking system represents the amount of information that can be embedded. Capacity can be assessed by calculating the ratio of capacity to reliability. This can be done

empirically by fixing one parameter (e.g. payload size) and determining the other parameter (e.g. error rate). Those results can then be used to estimate the theoretical maximum capacity of the watermarking system under consideration. Since an application ultimately identifies maximum capacity requirement, it may appear that an excess capacity which exceeds an application requirement is not important, and therefore it may not be necessary to estimate a maximum theoretical capacity of the watermarking system under consideration. However, the excess capacity is important because it can always be traded for improvements in reliability. This can be done by using the excess payload bits for error detection and/or correction.

Another property to take into account is the watermark access unit or granularity. It represents the smallest part of an audiovisual signal needed for reliable detection of a watermark and extraction of its payload. In the case of image watermarking, this property can be evaluated by using test images of different sizes.

In general, in order to obtain statistically valid results, it is important of ensure that:

- the watermarking system under consideration is tested using a large number of test inputs,
- the set of test inputs is representative of what is expected in the operating environment (application), and
- tests are executed multiple times using different watermarking keys.

11.10.3 Benchmarking

There are a number of benchmarking tools that have been created to standardize watermarking system evaluating processes.

Stirmark is a benchmarking tool for digital watermarking designed to test robustness. For a given watermarked input image, Stirmark generates a number of modified images that can then be used to verify if the embedded watermark can still be detected. The following image alterations have been implemented in Stirmark Version 3.1: cropping, flip, rotation, rotation-scale, sharpening, Gaussian filtering,

random bending, linear transformations, aspect ratio, scale changes, line removal, color reduction, and JPEG compression. More information about Stirmark can be found at: www.watermarkingworld.org.

Checkmark is a benchmarking suite for digital watermarking developed on MATLAB under UNIX and Windows. It is recognized as an effective tool for evaluating and rating watermarking systems. Checkmark offers some additional attacks not present in Stirmark. Also, it takes the watermark application into account, which means that the scores from individual attacks are weighted according to their importance for a given watermark purpose. Checkmark has implemented the following image alterations: wavelet compression (JPEG 2000 based on Jasper), projective transformations, modeling of video distortions based on projective transformations, warping, copy, template removal, denoising (midpoint, trimmed mean, soft and hard thresholding, Wiener filtering), denoising followed by perceptual remodulation, nonlinear line removal, and collage. Additional information about Checkmark can be found at: watermarking.unige.ch/Checkmark/

Optimark is a benchmarking tool developed to address some deficiencies recognized in Stirmark 3.1. Some of its features are: graphical user interface, detection performance evaluation using multiple trials utilizing different watermarking keys and messages, ROC curve, detection and embedding time evaluation, and payload size evaluation. Optimark information can be found at: poseidon.csd.auth.gr/optimark.

Certimark is a benchmarking suite developed for watermarking of visual content and certificating watermarking algorithms. It has been created as a result of a large research project funded by the European Union. Further details on Certimark can be found at: www.certimark.org.

Chapter 12

DIGITAL WATERMARKING AND BINARY IMAGES

In our society, documents represent a primary form of written communication, and large volumes are exchanged daily. Document recipients should be able to authenticate documents as well as identify document owners, and digital watermarking can be specifically used for that purpose. While many techniques have been proposed for watermarking of gray scale and color images, those techniques cannot be directly applied the binary images for a number of reasons. Gray scale and color image pixels take a wide range of values, and watermarking techniques typically make small modifications to the color or brightness values of the selected set of pixels without causing visually noticeable image distortion.

Binary images have only two distinct pixel color values. Therefore, it is not possible to make a small modification of those values, the approach that works so successfully with gray scale or color images. It is also not possible to apply a frequency domain approach, such as a spread spectrum embedding, to binary document image watermarking because of the need for post-embedding binarization of a watermarked image. Post-embedding binarization, to ensure that the marked image is still a two-color image, has been shown to create a perceptible distortion along the black-white boundaries and to disturb the embedded watermark to the point of removing it completely.

Binary images are images with only one bit of gray level resolution. Since only one bit is assigned to each image pixel, each binary image pixel has two possible gray level values, '0' and 1'. These two values are usually interpreted and displayed as two extreme gray tones, black and white, but there is no established convention that maps values '0' and '1' to 'black' and 'white' respectively. For example, the MathWorks's MATLAB product assigns 'black' to value '0' and 'white' to value '1', but it is not unusual to see those assignments reversed.

In other words, it is up to a user to assign a physical significance to the values '0' and '1'.

Binary images can be divided into two broad categories, *half-tone* and *document* images. This classification is based on characteristics of distribution of image pixels, which in turn depends on how an image is created. Half-tone images are created from gray scale images using *half-toning*, a process which takes a gray scale image and converts it into a binary image so that the original and the half-tone image appear similar when observed from a distance. Half-tone images are created for simplified processing or printing, and they can be found in printed materials, such as books, magazines, newspapers, and printer outputs. Document images are scanned representations of two-color documents, such as legal documents, birth certificates, digital books, engineering maps, architectural drawings, road maps, music scores, etc. This chapter will focus on issues related to document images.

Watermarking techniques for binary document images have some special requirements. For example, it is not possible to arbitrarily choose the set of pixels to modify in binary document images, because changing even a single white pixel to black in an all white section of a binary document image will produce a visible image distortion. This can easily be seen in Figure 12.1, which shows three images, the original image and two modified versions of the original image. Both modified versions have the same number of pixels changed. Modifications are made randomly in the Modified Image 1, and the Modified Image 2 is created from the original image by flipping two rows of white pixels adjacent to the black pixels on the two horizontal bars. Even though both modified versions have the same number of changed pixels, the distortion appears more pronounced in the Modified Image 1 where the pixels have been randomly selected for modification. This also demonstrates that the number of modified pixels in an image is not a good measure of visible image distortion, and that careful selection of pixel candidates for modification minimizes visible image distortion.

Since watermarking techniques for gray-scale and color images are not applicable to binary images in general, and binary document images in particular, different watermark embedding techniques need to be developed for those types of images. Until recently, there has been very little work published on watermarking techniques for

binary images. Chen et al. [10] presents a survey of techniques for data hiding in binary and text images, and classifies watermarking techniques based on the embedding method into the following categories:

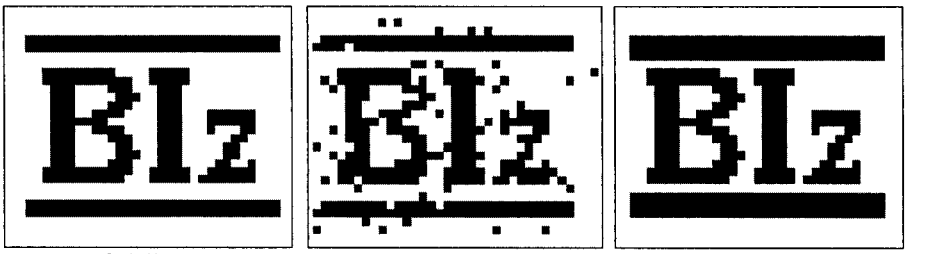


Figure 12.1. Original binary image and two modified versions of the original image with the same number of modified pixels. This demonstrates that the number of modified pixels in an image is not a good measure of visible image distortion. Distortion is much more visible on the Modified Image 1 than on the Modified Image 2.

- Text, line, word or character shifting,
- Modifications of character features,
- Modifications of run-length patterns,
- Fixed partitioning of an image into blocks,
- Boundary modifications,
- Modifications of half-tone images.

The text line, word, or character shifting category of watermarking techniques embeds data into documents containing formatted text by changing line, word, or character spacing by a very small amount, e.g. 1/150 in. [52]. For example, a bit '1' can be embedded by shifting a selected text line, called the marked line, up to make the space between it and the control line above slightly smaller, and the space between it and the control line bellow slightly larger. A bit '0' can be embedded by shifting the selected text line down to make its distance to the control line above greater and to the control line below smaller. Equivalently, words and characters within a text line can be shifted left or right, to make the space between words or

characters smaller or larger depending on whether a '1' or '0' needs to be embedded. Line shifts representing the embedded data are detected by first identifying the control lines, and then verifying the distance between the marked line and two control lines, which are located immediately above and bellow the marked line.

Another category of watermarking techniques is based on extracting text character features, and modifying them to embed data [2]. One such technique uses an analysis of connected components in a document to initially identify text areas. The text areas that are close to each other are grouped together, and each group is divided into four partitions, which are assigned to two sets. The average width of the horizontal strokes of characters is computed then used as the embedding feature. Data is embedded by manipulating the width of the character's horizontal strokes within the partitions. For example, to embed '1', the width of horizontal strokes is increased in partitions belonging to the first set, and decreased in partitions belonging to the second set. To embed '0', the opposite operation is used, and the width of horizontal strokes is decreased in the first set and increased in the second set. Two partitions are used for each set to help increase detection probability. The watermark detector divides the test document into groups and then it partitions those groups, the same way the watermark embedder does it. The horizontal stroke width is calculated for each partition within a group, and results belonging to the same sets are added together. The difference between the calculated widths of horizontal stroke in first and second set in each group is then compared to a threshold. The difference will be larger than the threshold if the stroke widths were increased in partitions belonging to the first set and decreased in partitions belonging to the second set. Therefore, if the difference is larger than a threshold, then '1' is detected. The difference will be smaller than a negative threshold if the stroke widths were decreased in partitions belonging to the first set and increased in partitions belonging to the second set. Therefore, if the difference is smaller than the negative threshold, then '0' is detected.

Facsimile images are two color images that have 2376 lines, with each line having 1728 pixels yielding 4,105,728 pixels per image. Those pixels are not coded as individual bits. Instead they are coded using *run length coding*, where lengths of runs of black and white pixels are coded using entropy coding based on statistical distributions of lengths of bit runs. There exists a category of

watermarking techniques, which embeds data by manipulating the patterns of run lengths, and those techniques are suitable for watermarking of facsimile images. For example, signature data bits can be embedded into a facsimile image by modifying run lengths and replacing the rightmost pixel of each line with the next bit of the signature [51].

These three categories of watermarking systems, which embed data by modifying the spacing between lines, words, or characters, or by modifying some character features, or by modifying run-length patterns, are applicable only to special kinds of two color images, namely text and facsimile images. The other categories of watermarking systems listed earlier, use data embedding methods that are more generally applicable to binary images, and they are outlined in Table 12.1.

Table 12.1. Classification of watermarking techniques suitable for binary images

Embedding Method	**Description**	**Applicable to**
Fixed Image Partitioning	Partitions an image into fixed-size blocks, and embeds data bits into individual blocks by enforcing some block based feature or property.	A generic binary image
Boundary Modification	Embeds data based on a pre-defined, fixed set of pairs of complementary (ie. dual) boundary patterns. Data bits are embedded by enforcing the appropriate pattern from the pair.	Document images with connected components, e.g. text documents or engineering drawings
Modifications of Half-tone Images	Embeds data bits directly into a randomly selected set of image pixels.	Half-tone images only, the binary images, which are perceived as gray-scale image when looked at from distance.

Half-toning is a process of converting a continuous tone image into an output image of two levels, so that the original and the half-tone image appear similar when observed from distance [69]. There are three popular half-toning techniques: ordered dithering, error diffusion, and optimization. The simplest technique is ordered dithering, which is based on a *dither matrix* whose values are used as thresholds for the corresponding image pixels. Error diffusion is computationally more intensive technique that generates higher-quality half-tone images suitable for low- to medium-resolution devices. Optimization techniques, while the most complex and computationally intensive, tend to generate the best quality half-tone images.

The main application of half-toning is printing and displaying images using devices capable of displaying only two colors. An example of a half-tone image generated using the ordered dithering technique is shown in Figure 12.2. The original Lena image is an 8-bit gray scale image, and the dither Lena image is a binary half-tone image. When observed from some distance, these two images appear to be almost identical, even though pixels in the original image can have one of the 256 possible gray levels, whereas pixels of its dithered counterpart can be either black or white. The zoomed in areas of the dithered image reveal that binary half-tone images do not have well defined boundaries between black and white areas because black and white pixels are well interlaced.

Several watermarking techniques have been proposed for half-tone images. Hel-Or [39] embeds data into an image as part of the image half-tone processing. The proposed method generates the half-tone image using two different dither matrices. Different dither matrices are used on different image areas, resulting in different areas of the output half-tone image having different statistical properties. Fu and Au [34] use two stochastic screen patterns to create two half-tone images. Data is embedded using correlation between two screens, and the embedded logo pattern can be viewed only after the two half-tone images are overlaid. Fu and Au also describe another data hiding technique suitable for half-tone images, called data hiding by self-toggling (DHST). This technique pseudo-randomly generates a set of pixels in an image to be used for data embedding. One bit is embedded in each location by forcing a specific pixel to be either black or white depending on whether '0' or '1' needs to be embedded. The detector uses the same random number generator

seed to create the sequence of pixels, which carry the message, and then it simply reads the values of those pixels. Kim and Afif [41] extend this technique into a cryptographically secure fragile authentication watermarking scheme.

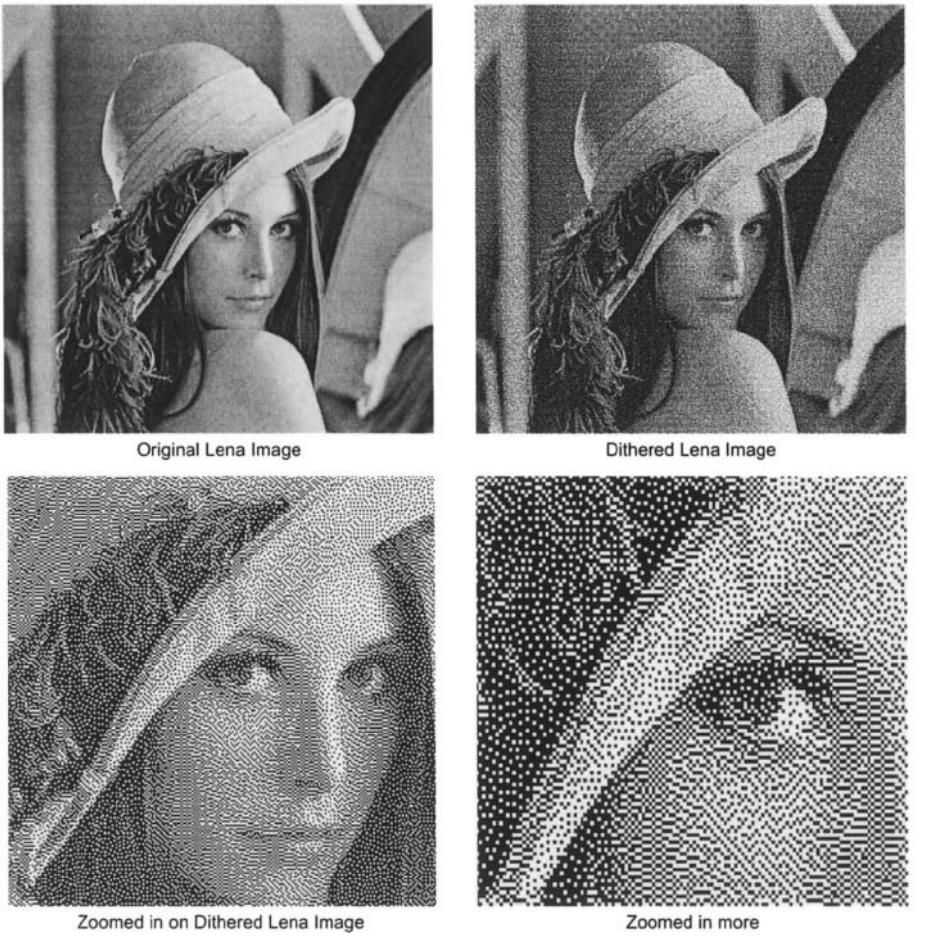


Figure 12.2. Original Lena image, and half-toned Lena image created using the ordered dithering.

The other type of binary image is the binary document image type, which is characterized by sharp contrast and clear boundaries between black and white areas (e.g. text document, engineering drawing, music score, maps etc.). Data embedding techniques based on modifying randomly selected image pixels which are

suitable for half-tone images, are not applicable to binary document images because random modifications of binary document image pixels will create visible salt-and-pepper noise. Pixel modifications in binary document images should be carried out at the boundary between black and white image areas, because modifications in those image areas will be less noticeable than modifications of randomly selected pixels.

Boundary modification-based watermarking schemes embed data into a binary document image by modifying boundary patterns of components in an image. These watermarking techniques can be used with images that have connected components, such as text characters or engineering symbols. One such scheme is presented in [53], where data is embedded in the 8-connected boundary of a character. This scheme is based on identifying five pixel boundary patterns that are common for text characters. Those five pixel patterns are paired together, so that patterns in each pair differ only in one pixel. The fixed set of pairs is then used for data embedding. Since patterns in each pair differ only in one pixel, modification of that pixel in one pattern of the pair results in the other pattern of the pair. The patterns in each pair are assigned values '0' and '1'. Data embedding is performed by finding patterns in the 8-connected boundary of image components that match the patterns from the set, and then either leaving the found patterns unchanged, or modifying one pixel to change patterns to their respective pair counterparts, depending on whether '0' or '1' needs to be embedded. The detection process simply searches for the patterns in an image and interprets those patterns appropriately. A duality of patterns in a pair allows for easy detection of the embedded data without using the original image.

A class of watermarking systems based on fixed image partitioning divides an image into fixed-size blocks and embeds one or more bits into individual blocks by modifying some pixels in that block in order to enforce a specific block-based feature. An example of such a scheme, called the Koch embedding, is proposed by Zhao and Koch [44]. An image is partitioned into 8×8 blocks, and data bits are embedded by modifying pixels within partitioning blocks to enforce a specific ratio between black and white pixels. This method embeds '1' by forcing the black-to-white pixel ratio to be larger than one, and conversely, it embeds '0' by forcing this ratio to be smaller than one. This embedding scheme has a problem dealing with blocks that have

either low or high percentage of black pixels. In order to embed a specific bit in those blocks, many pixels may need to be modified which can cause visually perceptible distortion of the document. This problem is solved by embedding data bits only into blocks with an acceptable black-to-white pixel ratio. As illustrated in Figure 12.3, only blocks having 25–75% black pixels may be used for embedding. To embed '0' into an embeddable image block, black pixels are turned into white until fewer than 40% of block pixels are black. To embed '1' into an embeddable image block, white pixels are turned into black pixels until more than 60% of block pixels are black. This embedding method achieves robustness, while limiting image distortion, at the expense of embedding capacity.

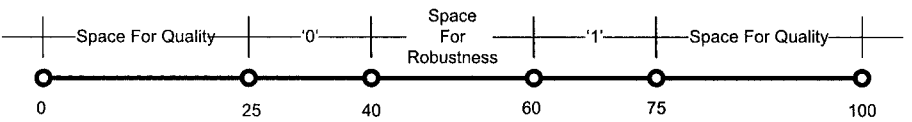


Figure 12.3. Rules for the data embedding based on a ratio of black pixels in an image partitioning block.

This embedding restriction helps reduce perceptible distortion of the watermarked document by reducing the number of pixels that need to be modified in order to enforce specific block-based feature. However, the achievable embedding capacity of this scheme is limited because it only allows bits to be embedded in a subset of image blocks.

The decision about which pixels within the block are modified as part of the Koch embedding scheme affects the quality of the watermarked image. The Koch watermarking scheme uses two different strategies for selecting pixel candidates for modification depending on whether the binary images are half-tone or document images. The strategy for half-tone images is based on an assumption that modified pixels should be evenly distributed throughout the whole block, and the selection algorithm searches for those pixels that have the most neighbors with the same value, either black or white. The strategy for binary document images targets black-to-white boundary pixels, and the pixel selection algorithm searches for those pixels that have most neighbors with the opposite value.

Funk and Schmucker [36] adopt Koch embedding and use it to implement a robust, high capacity watermarking scheme suitable for

music score images. This watermarking scheme, called the Wedelmusic, embeds the Wedelmusic object identifier, a 108-bit message, into images created by scanning music scores at 300 dpi. The Wedelmusic identifier is embedded redundantly to make it robust to changes in image resolution and standard image pixel operations such as erosion and dilatation.

The quality of images watermarked with Wedelmusic technique was measured using subjective tests based on the rating scheme proposed by the International Telecommunication Union (ITU). The ITU-R BT500 recommendations deal with methodologies for the subjective assessment of the quality of television picture. A set of test images of music scores is presented to a group of musicians, and they are asked to assign a quality rating (e.g. excellent, good, fair, poor, or bad) based on the level of impairment (e.g. imperceptible, perceptible but not annoying, slightly annoying, annoying, very annoying) they observe. The test images were created using different block sizes to embed data. The quality ratings identified the 4×4 block size as the optimal block size. Use of a bigger block size resulted in very visible distortion. Artifacts were still visible even with a block size of 4×4, but they were not annoying to musicians.

Min Wu, et al [80] proposes another watermarking method for digital binary images based on the fixed partitioning of an image into blocks. This method, called the MWLUT (for Min Wu's Look-Up Table), is designed for image annotation and authentication. The main innovation of the MWLUT watermarking scheme is the method for scoring image pixels, with an objective to identify the set of pixels whose modification, as part of watermark embedding, will cause the least visible distortion. Image pixels are scored according to their 3×3 neighborhoods. The score ranges from 0 to 1, with 0 assigned to those pixels that should never be modified, and with higher scores generally identifying pixels whose modification will introduce little visible distortion. The scoring is driven by subjective observations and estimates of what kinds of pixel modifications are more noticeable to human eyes. Additionally, the pixel scoring process attempts to preserve the smoothness of horizontal, vertical and diagonal lines in a local 3×3 window, as well as the number of black and white clusters in the 3×3 neighborhood. The scoring process results in a list of 512 elements representing scores of all possible 3×3 patterns indicating how noticeable the modification of the center

pixel will be. The list of scores is calculated once, and then it is stored in a lookup table to be used as part of data embedding operation.

To embed data bits into an image, all image pixels are first assigned scores from the lookup table based on the value of their 3×3 neighborhoods. The image is then divided into the set of fixed size blocks, and the pixel with the highest score in each block is selected for modification (flipping) as part of data embedding operation.

The MWLUT watermarking scheme uses a binary logo image as the watermark, and it embeds data bits representing the logo image into a block partitioned binary image, one bit per block. Data embedding is done by enforcing an odd or even number of black pixels in a block. For example, '0' is embedded in a block by modifying at most one block pixel to ensure that the number of black pixels is even. Conversely, '1' is embedded in a block by modifying at most one block pixel to ensure that the number of black pixels in a block is odd.

Image pixels with high scores, which represent good candidates for modification, are not evenly distributed across image partitioning blocks. Because of that, many image partitioning blocks will not have any good candidates for modification, and consequently those blocks will not be usable for embedding. This problem is resolved with a random shuffle of image pixels, which distributes good modification candidate pixels across image partitioning blocks more evenly. This shuffle increases the embedding capacity of the MWLUT watermarking method by increasing the number of blocks having at least one good modification candidate pixel.

The embedded bits form a fragile watermark that can be used for authentication of binary images and detection of image tampering. A watermark detector will successfully extract the embedded logo image as a proof that the document is authentic. However, if a watermarked image has been changed, a watermark detector will fail to extract the embedded logo image, and this failure indicates that the document was tampered with.

This method of embedding an authentication watermark into a binary image can be used only to prove that an image was not altered. If the watermark detector fails to extract the embedded logo

image, the only possible conclusion is that the test image is not authentic. However, failure to detect the fragile mark can have several causes. First, failure could be caused by illegal tampering with the intention to make a significant change in the conveyed meaning. This is the type of malicious image tampering an authentication watermarking scheme should identify. Failure could also be caused by standard image processing operations and manipulations, such as image compression, printing and scanning. These modifications are not significant in a sense that they do not change the content and meaning of an image, and consequently, the test image is still an authentic one. Finally, failure could happen if the original image was not marked at all. This situation occurs when the original image simply belongs to someone else, and no malicious and illegal image tampering is involved at all. The MWLUT method of embedding an authentication watermark into a binary image cannot distinguish the reasons for failing to detect the fragile mark.

This problem can be solved by embedding two watermarks into a binary image, one to convey binary image ownership information, and the other one to confirm image authenticity. The authentication watermark is embedded as a fragile mark as described above. It is used to prove authenticity if there were no changes made to the image. The ownership watermark is embedded as a robust mark capable of surviving various kinds of image alterations, so that it can be used to identify the cause of the failure to detect the fragile mark. With these two watermarks embedded in an image, one should always be able to extract the ownership information from an altered image. This information can then be used to contact the original image owner and request the original image version. Having access to the original image version makes it possible to detect whether illicit image tampering caused the authentication test to fail. Finally, the detector should fail to extract the robust watermark containing the ownership information only if the test image was not watermarked at all.

Chapter 13

TWO-LEVEL MARKS: DESIGN

Watermarking can be defined as embedding information-carrying signals, such as an ASCII string or a logo pattern, into a host signal, such as a binary image. The information-carrying signals are called watermarks, and they should be embedded without introducing visible distortion to the host signal. The two-level watermarking scheme presented here simultaneously embeds two watermarks into the host signal. These two watermarks serve different purposes. One is used for authentication verification and the other one for ownership information. The two watermarks used in the experiments presented in this document are logo patterns illustrated in Figure 13.1, and the host signal is a binary document image.

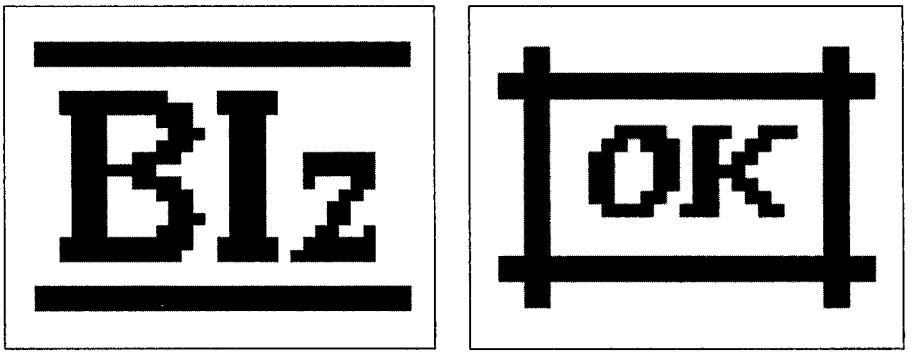


Figure 13.1. These two binary logo images are used as watermarks in the experiments presented in this document.

The first pattern is the 26×35 Biz logo image. This 910-bit message is intended to represent a company logo which will be embedded into a document image as a robust watermark. The watermark robustness refers to the ability of a specific watermarking scheme to detect and extract the embedded watermark after a standard signal processing procedure has been applied to the document image, with or without intent to prevent detection of the embedded watermark. The second pattern is the 26×35 OK logo image. This 910-bit

message will be embedded into a document image as a fragile mark which is destroyed if the document image is altered. The OK logo is intended to confirm document image authenticity.

Watermarks can be embedded into binary images a number of different ways. The embedding technique needs to be selected based on the characteristics and the representation of the binary image. For example, the class of binary images that use raster-scan representation is divided into two categories according to a distribution of black and white pixels. The two categories, half-tone and binary document image require different watermarking techniques

The half-tone binary image category is characterized by black and white pixels being well interlaced within an image, whereas the binary document image category includes images that have clearly defined boundaries between black and white areas. Since black and white pixels in a half-tone image are well interlaced, random modification of some of those pixels will not create visible image distortion. Because of that, watermarks can be embedded into half-tone images by changing values of randomly selected image pixels, so that individual data bits correspond to the values of those randomly selected image pixels, one data bit for each pixel. This data embedding scheme depends on both the embedder and the detector sharing the knowledge of what subset of image pixels is being used for data embedding, and in what order. This information is shared by ensuring that both the embedder and the detector use the same secret key as a seed to the random number generator that selects image pixels for embedding. This watermarking scheme is similar to the scheme that embeds data in the least significant bit (LSB) of each pixel in gray-scale and color images, except that instead of using the LSB, the pixels to be used for embedding are randomly selected.

Watermarking schemes based on modifying randomly selected image pixels are not suitable for binary document images because random modifications of document image pixels cause the annoying image distortions as illustrated in Figure 12.1.

Watermarking schemes based on fixed image partitioning are suitable for watermarking binary document images. As described earlier, this category of watermarking techniques is based on

partitioning an image into fixed-size blocks, and enforcing a certain block feature or property by modifying a small number of block pixels to embed one or more bits into each block. Pixels should not be selected and modified randomly. Instead, pixels should be selected for modification to minimize visible document image distortion.

The fixed image partitioning method is used for watermark embedding and detection as part of a general framework for the two-level watermarking scheme, as illustrated in Figure 13.2. This framework offers a logical separation of subcomponents which deal with how to:

- Select an initial set of pixel candidates for modification,
- Categorize, score, and/or filter pixel candidates to help minimize visible distortion caused by modification of image pixels,
- Select what block feature to enforce as part of data embedding operation.

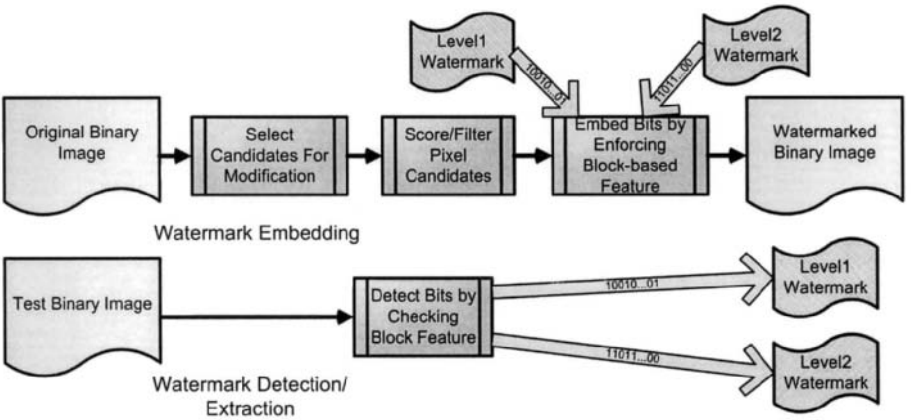


Figure 13.2. Block diagram of the watermark embedding and detection process.

The two-level watermarking scheme presented here is designed for binary document images only, and it is based on embedding two watermarks, one robust and one fragile. Individual watermarks are embedded into a binary image in an overlapped fashion, with the robust watermark being embedded first. The order of watermark

embedding is important. Embedding the fragile mark before the robust watermark will not work because the detection of the fragile mark will likely fail due to the interference caused by embedding the robust watermark.

The initial set of pixel candidates for modification is selected from the set of boundary pixels between black and white areas of an image. The selection process takes into consideration all white 8-neighbor pixels of the black boundary pixels, and all black 8-neighbor pixels of the white boundary pixels. The 8-neighbour pixels are horizontal, vertical and diagonal neighbors of a pixel. Those pixels are scored according to how noticeable the changes of those pixels are expected to be. This score is calculated using the Structural Neighborhood Distortion Measure (SNDM), an objective distortion measure designed for binary document images. The SNDM score is calculated for each modification candidate pixel, taking into consideration its $n \times n$ neighborhood, where n=3,5,7... Once the modification candidates have been selected and scored, they can be used to embed two watermarks by enforcing two different and independent block-based features.

This data embedding and detection method represents a new two-level watermarking scheme that can be used to hide a moderate amount of data, representing two distinct watermarks, in a general binary document image, such as scanned text, figure, map, drawing, music score, etc. The hidden data can be extracted without using the original image, and this watermarking scheme can be used to help identify the owner of a document should it fail the authentication test.

13.1 DISTORTION MEASURE

Modifications of binary image pixels from black to white and from white to black will cause image distortion. Given enough pixel candidates for modification, the objective should be to select and modify only those pixels whose modification will cause image distortion that is visually the least perceptible. The selection of best candidates for modification can be done using human observers. Those methods for distortion measure are called *subjective* methods because their results depend on the individual, subjective perceptions of the people involved in the distortion evaluation experiments. Subjective methods for measuring image distortion are

difficult to replicate and hard to incorporate into an algorithm, which selects image pixels whose modification will not be too noticeable. One example of subjective method used for pixel scoring is the MWLUT [80].

The MWLUT method scores modification candidate pixels based on a scoring table created by analyzing all possible combinations of black and white pixels in a 3×3 neighborhood. The score is assigned to every one of 512 possible patterns, based on a subjective estimate of the level of visually perceptible distortion that a modification of the center pixels of a specific 3×3 pattern will create. The 3×3 patterns are analyzed once. The assigned scores are stored in a look-up table and used by a data embedding algorithm to identify pixels whose modification will cause minimum visible distortion. The look-up table score is keyed based on the corresponding numerical value that the pixel's 3×3 neighborhood maps into. While the analysis of black and white patterns and the creation of a look-up table can be done for 3×3 patterns, the 3×3 neighborhood could be too small a neighborhood for an accurate assessment of visual distortion caused by modification of the center pixel. The scoring based on an analysis of larger patterns (i.e. 5×5, or 7×7) may produce better results, but an analysis of larger patterns is much more extensive and complicated, requiring the use of prohibitively large look-up tables.

The selection of best candidates for modification can also be done using objective methods that measure distortion. The traditional objective distortion metrics that are frequently used include the Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR).

The MSE is one of the simplest distortion measures. It examines the magnitude of difference between two images, pixel by pixel, in the form of the squared error of a pair of pixel intensities, and derives its measure as:

$$MSE = \sum_{i=1}^{N}\sum_{j=1}^{M}(a_{i,j} - b_{i,j})^2, \tag{16}$$

where a and b represent two images of resolution $N \times M$, and $a_{i,j}$ and $b_{i,j}$ are the intensities of the pixel (i,j) in a and b respectively.

The corresponding PSNR is defined as:

$$PSNR(dB) = 10\log_{10}\frac{P^2}{MSE}, \tag{17}$$

where P is the maximum pixel intensity. For example, P is 255 for an 8-bit image, and it is 1 for a binary image.

Since PSNR is based on the MSE, these two distortion measures are essentially equivalent. The problem with these distortion measures is that they do not provide a good measure of visible distortion. A good demonstration of that is exemplified by Figure 12.1. Both second and third images have been created from the first image by modifying the same number of pixels, which means that they both have the same MSE and PSNR. However, the distortions perceived by the human eye are quite different for the second and third images. The reason is that both MSE and PSNR measure distortion based on the state of individual pixels, without considering any structural information.

For binary images, the MSE actually represents the number of differences between two images, and the large number of different pixels does not always result in a large structural difference between two images. Since the main function of the human visual system (HVS) is to extract structural information from the viewing field, a distortion metric, which takes into consideration structural distortion will provide a better approximation of visible image distortion.

The Distance-Reciprocal Distortion Measure (DRDM) correlates better with visually perceived distortion in binary images than MSE and PSNR [50]. This measure is created because models of the HVS built for natural images are not well suited for binary document images and consequently the perception of distortion in binary document images is different than perception of distortion in natural images. One reason for the difference is that, in any particular language, people know very well what a certain symbol should look like. If modification of binary document image pixels changes the structural form of those symbols, the resulting binary document distortion could be very visually disturbing to a casual observer.

The DRDM is an objective measure of visible distortion between two binary document images. It is designed based on an assumption that a distance between two pixels within an image plays an

important role in how the HVS perceives the mutual interference of those two pixels. Modification of pixels is more visible if they are closer to the area of viewer's focus. The closer the two pixels are, the more sensitive the HVS is to the change of one pixel when focusing on the other. Additionally, when observing the eight neighbors of a pixel, the diagonal neighbors are considered to be farther away from the pixel than its horizontal and vertical neighbors. Consequently, when focusing on a specific pixel, modifications of its diagonal neighbors are expected to have less visual effect than modifications of its horizontal and vertical neighbors.

The DRDM method measures distortion between two binary images, a and b, using a normalized weight matrix W_m of size $m \times m$, where m=3,5,7..., with each of its weights representing reciprocal value of the distance measured from the center pixel. The distortion is calculated as:

$$d = \frac{\sum d_k}{K}, \tag{18}$$

where K is defined as the number of non-uniform (not all black or all white pixels) 8×8 blocks in the image a, and d_k is a local distortion calculated in the $m \times m$ neighborhood for each pixel of difference between image a and image b, using the following formula:

$$d_k = \sum_{mxm} [\,|\, a_m - b_m \,|\, \times W_m\,]. \tag{19}$$

The DRDM provides a distortion measure, which correlates well with subjective methods, and hence it is superior to the MSE and PSNR [50].

The two-level watermarking scheme presented here uses DRDM as an ultimate measure of visible distortion caused by embedding watermarks into binary images.

The Structural Neighborhood Distortion Measure (SNDM) is an objective metric designed for scoring of binary document image pixels. Its objective is to identify the best pixel candidates for modification; those pixels whose modification will cause the minimum visible distortion. The SNDM takes into consideration the $m \times m$, m=3,5,7,... neighborhood of an individual pixel, and it calculates the pixel's modification score in that neighborhood. A

modification score is a number between 0 and 1, where modification of a pixel with the highest score is expected to introduce the minimum visible image distortion. The SNDM corresponds well with the subjective methods because it favors pixel modifications that contribute to the creation of more compact structures or objects in a local neighborhood.

The SNDM scoring method is based on the reciprocal distance matrix D_m, for an $m \times m$ neighborhood. An example of the D_3 matrix is shown in Figure 13.3.

0.7071	1.0	0.7071
1.0	0	1.0
0.7071	1.0	0.7071

Figure 13.3. Reciprocal 3x3 Euclidian distance matrix

The SNDM for an individual modification candidate pixel is calculated in the $m \times m$ neighborhood N_m of the candidate pixel, *cp*, as a normalized correlation between *XOR(cp, N_m)* and D_m:

$$SNDM_{cp} = \frac{(cp\ XOR\ N_m) \bullet D_m}{|D_m|}. \tag{20}$$

Pixel candidates for modification in a binary document image are not selected randomly. They are selected from the set of boundary pixels between white and black areas. The set of pixel candidates includes the white pixels, which have black pixel neighbors, and the black pixels, which have white pixel neighbors. The value of the candidate pixel, *cp*, is exclusively *OR*ed with pixels in its neighborhood, N_m, to ensure that correlation calculation depends only on the neighboring pixels that have different color than *cp*. In other words, pixels which have more neighbors of the opposite color are better candidates for modification than pixels which have more neighbors of the same color.

Figure 13.4 shows an example of two modification candidates in a 3×3 neighborhood. The first modification candidate is a white pixel with three neighboring black pixels as depicted in the image (a), and the second one is a white pixel with five neighboring black pixels, as depicted in the image (c). The first modification candidate has SNDM=0.3536 and the second one has SNDM=0.6063. If an image block has two 3×3 patterns in it, (a) and (c), it will have at least two pixel candidates for modification. However, if only one pixel needs to be modified, then, based on the respective pixel modification scores, the (c) candidate should be selected for modification because modification of the (c) candidate pixel is expected to cause smaller visually perceptible distortion than modification of the (a) candidate pixel. The images (b) and (d) show how the respective neighborhoods look, after both candidates have been modified.

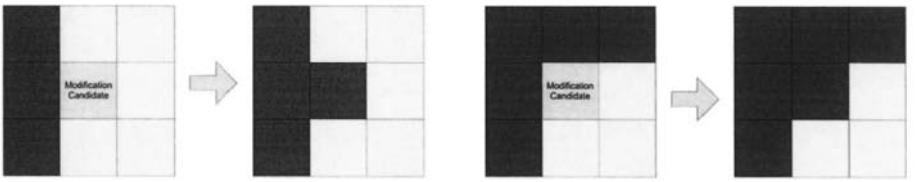


Figure 13.4. Images (a) and (c) represent two patterns in a 3x3 neighborhood. The modification candidate pixel is the central pixel in the neighborhood. The two images (b) and (d) represent new patterns created by modifying the modification candidate pixels.

13.2 IMAGE PARTITIONING

After identifying and scoring pixel candidates for modification, the image is partitioned into blocks, so that one or more data bits can be embedded in each block by modifying some of the modification candidate pixels in that block, in order to enforce a specific block-based feature. It is obvious that such a data embedding scheme works only if most of the blocks contain some number of modification candidates. Therefore, it is necessary to analyze a good representative set of binary document images to determine distribution of modification candidates across image partitioning blocks. The eight CCITT binary document images [31], scanned at 200 dpi, represent various types of binary document images. They include printed text documents using English, French and Chinese alphabet symbols, handwriting, hand drawing, tables, graphs, and

combination of text and graphs. Figure 13.5 shows snapshots of those eight CCITT images.

The eight CCITT images are partitioned into blocks using four different block sizes, 8×8, 16×16, 32×32 and 64×64, and the distribution of the number of modification candidates is calculated in each case. The results plotted in Figure 13.6 demonstrate that, irrespective of what block size is selected, more than 70% of blocks have zero candidates for modification, which means that more than 70% of blocks are not suitable for data embedding.

This result is not surprising, because binary document images in general have a lot of white areas, and modification candidate pixels are concentrated around black and white boundary areas.

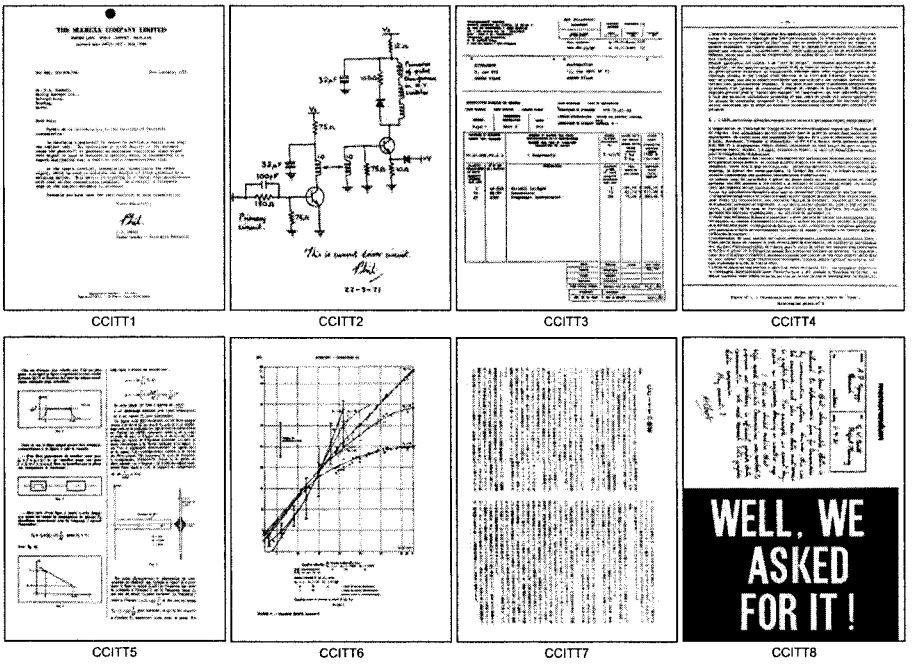


Figure 13.5. Eight 2376x1728 binary document images, CCITT1-CCITT8, scanned at 200 dpi.

Modification candidate pixels have to be distributed across image partitioning blocks more evenly, and this can be achieved by doing a key-based permutation of image pixels. A key-based permutation is

a pseudo-random permutation where a permutation key is the seed to the pseudo-random number generator. This is depicted in Figure 13.7.

In addition to ensuring a more even distribution of modification candidate pixels across image blocks, a key-based permutation of image pixels provides security for the embedding scheme. Without the knowledge of the key, it is not possible to determine to which block any specific modification candidate pixel belongs. It also ensures that any illicit modification of image pixels will not affect a feature of a single image block, but rather that it will affect the features of many blocks. Consequently, many embedded data bits, instead of one or few, will not be detected correctly, indicating document tampering.

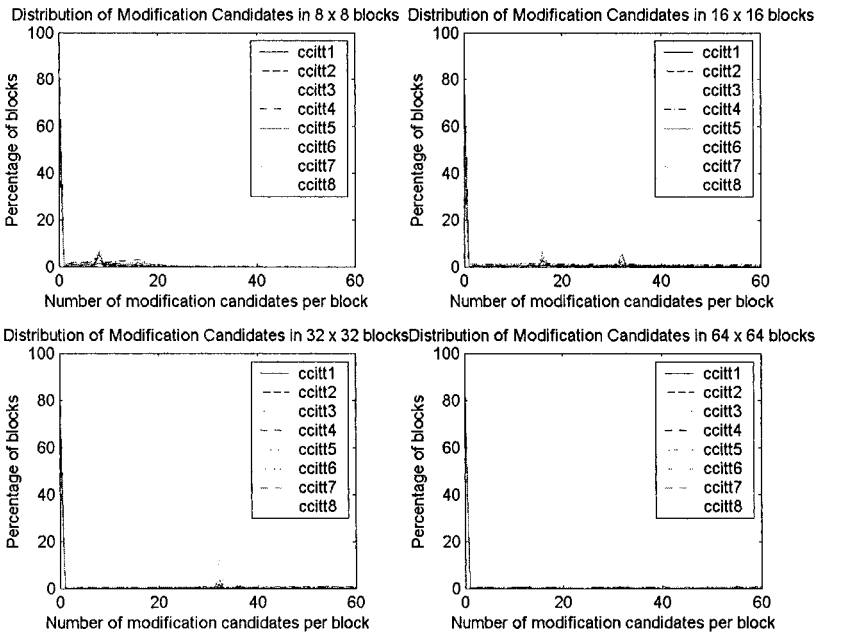


Figure 13.6. Distribution of modification candidate pixels in the eight CCITT binary document images for different block sizes

A key-based permutation of a vector with N elements can be easily created by generating and sorting N pseudo-random numbers, and using the sorting index as the resulting permutation. This is the

approach the MathWorks, Inc. used to implement the MATLAB *randperm* function. For example, a pseudo-random permutation of the vector v can be created as illustrated in Table 13.1. First generate the sequence r_6 of 6 pseudo-random numbers. Sorting the r_6 yields both the sorted vector sr_6, and the sorting index I. The sorting index, I, maps r_6 to sr_6, i.e. $sr_6 = r_6(I)$, and it represents a key-based permutation of v. More generally, if $\{r_k\}$ represents a sequence of N pseudo-random numbers, the $\{sr_k\}$ is its sorted representation, and the sorting index, I, maps $\{r_k\}$ to $\{sr_k\}$, then a pseudo-random permutation, *v_perm*, of the vector v of N elements, is given by $v\_perm = \{v(I)\}$. Since the sorting algorithm has complexity of *O(NlogN)*, the complexity of this algorithm for creating a pseudo-random permutation of image pixels is *O(NlogN)*, where N is the number of pixels in the image.

Table 13.1. Vector I represents a pseudo-random permutation of the vector v. Vector I is the sorting index of the pseudo-randomly created vector *r6*.

v	1	2	3	4	5	6
r6	0.9501	0.2311	0.6068	0.4860	0.8913	0.7621
sr6	0.2311	0.4860	0.6068	0.7621	0.8913	0.9501
I	2	4	3	6	5	1

Since images in general have millions of pixels, an algorithm which has the complexity of *O(NlogN)* can take a great deal of time to complete. Fortunately, it is possible to improve this algorithm for calculating random permutation of the vector v with N elements. This improved algorithm creates random permutation by reordering vector elements in place [43][30]. The basic idea of this algorithm is as follows: Given a vector v of the size N, visit each consecutive vector element $v(i)$ starting from the last element $i=N$, and going towards the first element. For each visited element $v(i)$, randomly select an element index $j \in [1, i]$ and swap $v(i)$ and $v(j)$. This algorithm has complexity of *O(N)*, and it creates a uniform distribution of vector permutations.

Simulation results provided in Table 13.2 list the execution times of the two vector permutation algorithms. The improved algorithm

creates a random permutation of N elements more quickly than the MATLAB *randperm* function, and as N is increasing, the improvement ratio is increasing. The improvement ratio numbers clearly demonstrate the benefits of designing and using an *O(N)* algorithm over an *O(NlogN)* algorithm.

Table 13.2. Execution times in seconds for the two algorithms and the improvement ratios for different vector sizes.

N	10^4	10^5	10^6	10^7
Elapsed Time for *randperm* algorithm	0.0254	0.3486	6.0060	106.50
Elapsed Time for Improved algorithm	0.0039	0.0390	0.4775	6.0059
Improvement ratio	6.5128	8.9385	12.578	17.733

The plots on Figure 13.7 indicate that even after executing key-based image permutation, the 8×8 image partitioning still leaves a big portion of blocks without modification candidates. One extreme case is the CCITT2 document, a hand-drawn electrical circuit document image. Compared to the other documents, the CCITT2 document has relatively few modification candidates, so it is not surprising that more than 50% of its blocks have no modification candidates at all. On the other end of the spectrum are the CCITT4 and CCITT7 documents. They have many boundary areas and consequently, they have more modification candidate pixels. After performing a key-based permutation and 8×8 block partitioning of these two document images, approximately 20% of all the blocks have 5 modification candidate pixels, only 1% of blocks have no modification candidate pixels, and most of the blocks have between 1 and 10 modification pixels available. As the image partitioning block size increases, the number of modification candidate pixels per block increases as well. However, the overall data embedding capacity decreases because, as the block size increases, the number of blocks per image decreases. The plots of Figure 13.7 indicate that the 32×32 partitioning block size is optimum for the set of CCITT document images because all document images, including even the CCITT2, have no image partitioning blocks with zero modification candidate pixels.

The next section presents the analytic study of results of the random permutation of modification candidate pixels. Let I be a set of pixels in an image, and let MC be a subset of I consisting of pixels, which are candidates for modification. The set I is partitioned into non-overlapping subsets or blocks, B_i, so that each block has the same number of pixels. The total number of blocks in I is given by $N_B = \frac{|I|}{|B_i|}$, where $|I|$ and $|B_i|$ represent the number of pixels in I and B_i respectively. Assume that pixels I are merely pixel placeholders, so that the MC pixels can randomly be distributed into the pixel placeholders I, one MC pixel per pixel placeholder. Once MC pixels have been distributed, calculate what fraction of blocks will have exactly k MC pixels, for $k=0, \ldots |B_i|$. If m_k represents the number of blocks having exactly k MC pixels, then the fraction of blocks having exactly k MC pixels is equal to $\frac{m_k}{N_B}$, and its mean value can be calculated as [45]:

$$E\left[\frac{m_k}{N_B}\right] = \frac{\binom{|B_i|}{k}\binom{|I|-|B_i|}{|MC|-k}}{\binom{|I|}{|MC|}}, \quad k = 0,1,\ldots,|B_i|, \tag{21}$$

where the denominator $\binom{|I|}{|MC|}$ represents the number of ways one can distribute the $|MC|$ pixels into the $|I|$ pixel holders, and the numerator $\binom{|B_i|}{k}\binom{|I|-|B_i|}{|MC|-k}$ represents the number of ways k modification candidate pixels can be distributed within $|B_i|$ pixel place holders, while the rest of the modification candidate pixels, $|MC|-k$, are distributed throughout the rest of the pixel placeholders.

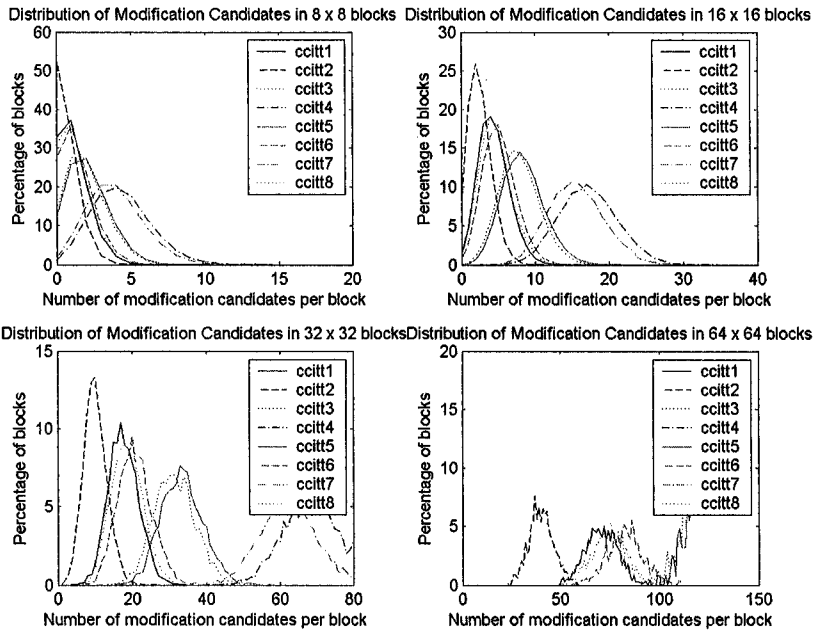


Figure 13.7. Distribution of modification candidate pixels in the eight CCITT binary images after random permutation of image pixels

The equation (6) shows that distribution of modification candidate pixels per block, after performing random permutation of image pixels, does not depend on the distribution of modification candidate pixels before random permutation. Instead, it only depends on (a) the ratio of the number of candidate pixels to the number of image pixels, and (b) the block size. This can be seen in Figure 13.8. If the image partitioning blocks are small, then in order to have no blocks with zero modification candidates after performing random permutation of image pixels, the original image needs to have a large fraction of modification candidate pixels. As the size of image partitioning blocks increases, a random permutation of a smaller fraction of modification candidate pixels will result in all blocks having at least one modification candidate available. This supports the experimental results presented earlier.

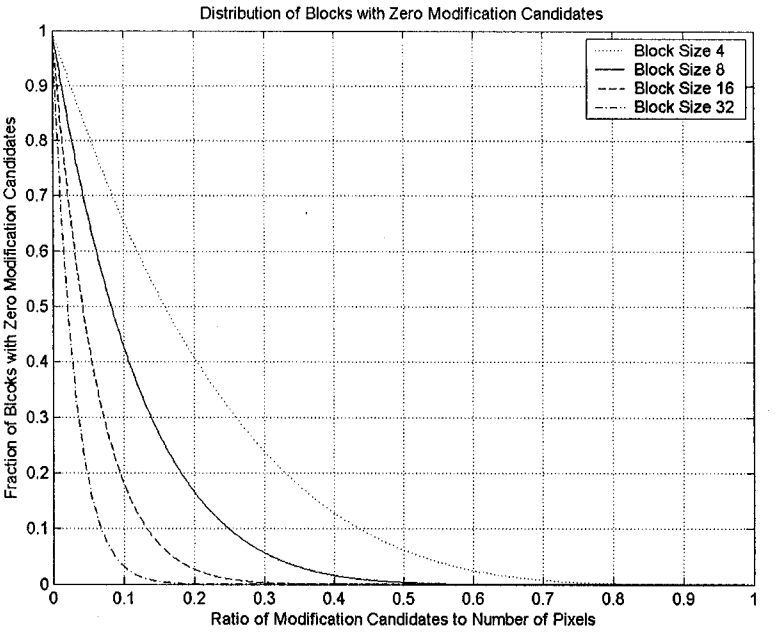


Figure 13.8. Fraction of blocks having 0 modification candidates for a 1024 pixel image, as the number of modification candidate pixels increases. Four curves represent distributions based on four different block sizes. If the blocks are small (4 pixels), 70% of image pixels need to be modification candidate pixels in order for the random permutation of image pixels to result in all the blocks having at least one modification candidate pixels. If the blocks are large (32 pixels), only 15% of image pixels need to be modification candidate pixels.

13.3 BLOCK FEATURE AND DATA EMBEDDING

The two-level watermarking scheme described here embeds data bits into a binary document image by first partitioning the image into blocks of the same size, and then changing values of the subset of image pixels in a block in order to enforce a specific block-based feature. In general, multiple bits can be embedded in each block by enforcing a block feature that has a number of different outcomes. For example, the data embedding technique called even/odd embedding [80], is based on forcing the number of black pixels in a block to be either odd or even depending on whether bit '1' or bit '0'

needs to be embedded. Since the number of black pixels in the block can either be odd or even, this feature has two possible outcomes, and therefore it can be used to embed a single data bit. An example of a feature with more than two outcomes is a feature that enforces relation via modulo operation [68]. In its simplest form, the feature based on the number of black pixels in a block *modulo-4* has four possible outcomes and therefore it can be used to embed 2 data bits per block. More generally, in order to be able to embed r bits per block, the feature to enforce needs to have 2^r different and clearly distinguishable outcomes.

As stated earlier, random selection of pixels to modify to enforce specific feature as part of data embedding often results in a visible and annoying image distortion. To avoid that, only a subset of block pixels called the set of modification candidate pixels should be used for data embedding. This subset consists of pixels whose modification will cause minimum visible image distortion.

In order to be able to embed data bits into blocks, depending on the selected embedding feature, each block has to have one or more modification candidate pixels available. For a simple even/odd embedding, each block has to have at least one modification candidate pixel available, and for the modulo-4 embedding each block has to have at least three modification candidate pixels available. But as shown earlier, binary document images may have more than 80% of blocks with no modification candidate pixels available. The key-based permutation of image pixels is used to redistribute modification candidate pixels throughout the image, so that a distribution of modification candidates per block is more even. While random permutation of pixels helps reduce the average number of blocks with no modification candidates to zero, the number of modification candidates per block within an image still varies from block to block, and it is still possible to have some image blocks with no modification candidate pixels available.

13.4 DATA EMBEDDING RATE

Image blocks containing a large number of modification candidate pixels can carry or hide a large number of data bits, and image blocks with only few modification candidates can hide perhaps only one data bit. Since the number of modification candidates per block within an image varies from block to block, it is possible to design an

image watermarking system which changes the data embedding rate from one image block to the other, embedding more data bits into the blocks which have more modification candidates available. Such a system is called a Variable Embedding Rate (VER) watermarking system. Its objective is to maximize data embedding capacity by embedding as many data bits in each block as possible. A VER system has to add some additional information into each block to indicate how many data bits have actually been embedded. This additional information is also known as control information or side information, and it represents an overhead of the system because it wastes data embedding capacity. A VER system is useful and efficient only if the number of data bits that can be embedded in each image block is large, so that the percentage of wasted data embedding capacity is small.

Another alternative is to embed the same number of bits in each image block. Such a system is called a Constant Embedding Rate (CER) watermarking system. A CER system does not need to waste block embedding capacity with side information detailing the number of embedded bits because the watermark detector already knows how many bits have been embedded in each block. However, CER systems waste embedding capacity in those blocks that can hide more data bits. Additionally, each block has to have enough modification candidates available to be able to hide the required number of data bits.

Experiments with the set of CCITT images demonstrate that binary document images on average do not have a large enough number of modification candidates per block to justify the overhead of the VER system. Based on this finding, binary document image watermarking schemes should use a constant embedding rate.

13.5 OPTIMIZATIONS

Random permutation of binary document image pixels distributes modification candidate pixels across image partitioning blocks more evenly. However, it is still possible to have some small number of blocks without enough modification candidates available to support required data embedding rate. This problem can be handled using the following data embedding optimizations.

13.5.1 Adaptive Random Permutation

A random permutation of binary document image pixels is based on a key, which is shared between watermark embedder and watermark detector. The embedder applies random permutation to an image before partitioning it into blocks, in order to ensure more even block-based distribution of modification candidate pixels. Data embedding is performed in that space by modifying candidate pixels to enforce certain block-based features. After that, a reverse permutation is performed to restore the image. The restored image is a watermarked version of the original image. The detector has to use the same key to perform random permutation of the watermarked image. It then partitions the image into the blocks, and extracts data bits from each block by verifying the state of the block's feature.

This watermarking scheme can be implemented to use the same shared secret key for all images. After all, it has been demonstrated earlier that distribution of modification candidate pixels per block after random permutation is independent of distribution of image modification candidates before permutation. Distribution depends on the block size and on the percentage of modification candidates in the whole image. However, it is still possible that, given an image, some specific instance of random permutation would result in a more uniform distribution of modification candidates per block than another instance of random permutation. In other words, two different random permutations of the same image could have a different number of blocks without enough modification candidates available to support required data embedding rate. The blocks which do not have enough modification candidates to support required embedding rate are called *non-embeddable* blocks, and data bits stored in those blocks will not be detected correctly. It is clear that the permutation that results in minimum number of non-embeddable blocks is the optimum permutation for data embedding into a specific binary document image.

A watermarking scheme requires both the embedder and the detector to use the same permutation key. If the key is fixed, the solution is trivial. However, if the embedder can use more than one permutation key, how will the detector know which permutation key was used to embed a watermark? One possibility is to create a secret list of permutation keys to share between the watermark embedder and the detector. For any given binary document image,

the embedder will try all the keys from the list, then select and use the one which creates a permutation with the minimum number of non-embeddable blocks. While the detector has the same secret list of keys in its possession, it does not know which specific key was used for data embedding, and without that information, it cannot detect and extract the embedded data. The information about what key to use cannot simply be embedded in the image as side or control information for the detector to use, because the detector needs to have the key in order to be able to extract any information.

This problem is similar to the classic data communication problem where two communicating parties have to agree on common capabilities, such as communications speed, before they can exchange any data. The communicating parties can use the training sequence to synchronize the initial data communication speed. The training sequence is a known pattern that transmitter sends at certain speed. A receiver will sample received data at different rates, until it finds the sampling rate that produces the expected data pattern. This data pattern can either be a known, pre-defined pattern, or it can be designed to be self-verifiable. The latter choice offers more flexibility because transmitters and receivers do not have to share data pattern information, and it can be easily implemented using some kind of message digest function.

An equivalent of the training sequence can be used in image watermarking to help the detector determine what random permutation key to use to detect and extract data bits. The first 12 embedded data bits, for example, could represent the training sequence pattern, consisting of 9 random bits and 3 bits representing some kind of digest of the first 9 bits (e.g. *XOR* of three sets of tree bits of the 9-bit pattern). The detector will try all random permutation keys from its secret list of keys, until it finds the permutation that accurately decodes the first 12 bits.

A random permutation with the wrong key will result in detection of data bits which are independent from each other, and which could take either value '0' or value '1' with the same probability, 2^{-1}. The probability of matching the 12-bit pattern using the wrong permutation key is then 2^{-12}.

13.5.2 Adaptive Image Partitioning Block Size

Some images with a low percentage of modification candidate pixels will have a large number of non-embeddable blocks irrespective of how many different random permutations one tries. Basically, given an image with small number of modification candidate pixels, no matter how the candidates are distributed across a fixed number of blocks, the average number of modification candidates per block is going to be small. In that case, the only way to increase the number of modification candidates per block, and consequently decrease the number of non-embeddable blocks, is to increase the block size. However, since both the embedder and the detector have to use the same block size, the block size used by the embedder has to somehow be communicated to the detector. This information can be passed to the detector the same way the information about the random permutation key is passed. In other words, the detector can figure out the block size indirectly using the training sequence approach, as an extension to detecting the permutation key. For example, the secret, shared list of permutation keys can be expanded with the information about the acceptable block sizes. The detector will try all combinations of random permutation keys and block sizes until it accurately detects the 12-bit data pattern.

13.5.3 Error Correction Coding

If after selecting the best random permutation and the best partitioning block size for a given image, the image still has non-embeddable blocks, some number of watermark bits will not be embedded, and the detector will not detect those bits correctly. That situation can be handled using Error Correction Coding (ECC). Basically the original watermark message can be encoded using ECC before it gets embedded in an image. After applying all optimizations described above, the number of non-embeddable blocks is expected to be very small, and consequently only a little error correction capability is needed.

13.6 UNIFORM QUANTIZATION BASED EMBEDDING

Watermarking based on Uniform Quantization [13][28] is a CER, block-based watermarking scheme that embeds one bit per image block using the uniform quantization approach. Uniform quantization is based on selecting a fixed quantization step size Q,

and forcing the total number of black pixels in a block to be either $2kQ$ or $(2k+1)Q$, for some integer k, in order to embed '0' or '1'. The mapping of data bits to a specific block feature, either $2kQ$ or $(2k+1)Q$, can be fixed, so that, the number of black pixels in a block is forced to be $2kQ$ in order to embed '1', and conversely, the number of black pixels in the block can always be forced to be $(2k+1)Q$ in order to embed '0'. Security of this data embedding scheme can be improved by mapping data bit into a specific feature based on a pseudo-randomly-created lookup table, as shown in Figure 13.9.

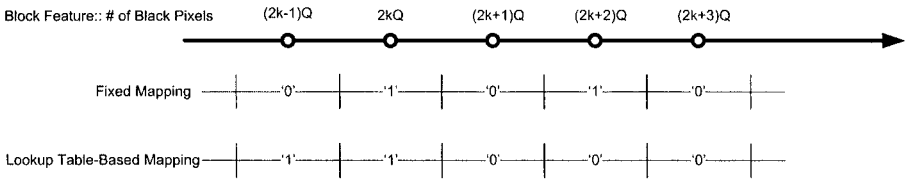


Figure 13.9. Illustration of Uniform Quantization based embedding. Bit mapping can either be fixed, or it can be based on a pseudo-randomly created lookup table. One bit is embedded in a block by forcing the number of black pixels to a value that matches the bit to be embedded using a defined mapping. A larger Q gives higher robustness against noise because the modification of the number of black pixels smaller than $Q/2$ will not affect the decoding accuracy.

The quantization step size Q determines the robustness of the embedding scheme against noise and attacks, and a larger Q provides higher robustness because more than $Q/2$ black pixels need to be modified in order to cause a decoding error. If $Q=2$, then the modification of a single pixel in a block will cause a decoding error, whereas if $Q=10$, at least five pixels need to be modified to cause a decoding error.

Detection is done by checking the enforced relationship in all blocks of a test image. To extract the bit b_i, the feature of the i^{th} block is verified. The number of black pixels in this block will define the quantization bin k, and selected mapping will indicate whether the bin k corresponds to '0' or '1'.

More formally, assume that the watermark message m represents a sequence of N bits, $m=[b1,b2,...bN]$. The N-dimensional uniform quantization table T^N is structured as a product of identical, one-dimensional, component uniform quantization table T^1, $T^N = T^1 \circ T^1 \circ ... \circ T^1$. The component quantization table T^1 has to be

separated into two distinct parts, T_0^1 *and* T_1^1, to allow the embedding of a single bit which could have one of two values '0' or '1'. The separation $T^1 = T_0^1 \cup T_1^1$ can be set as follows:

$$\begin{aligned} T_0^1 &= \{t = kQ \mid k = 0,1,2,...\} \\ T_1^1 &= \left\{t = kQ + \frac{Q}{2} \mid k = 0,1,2...\right\} \end{aligned} \tag{22}$$

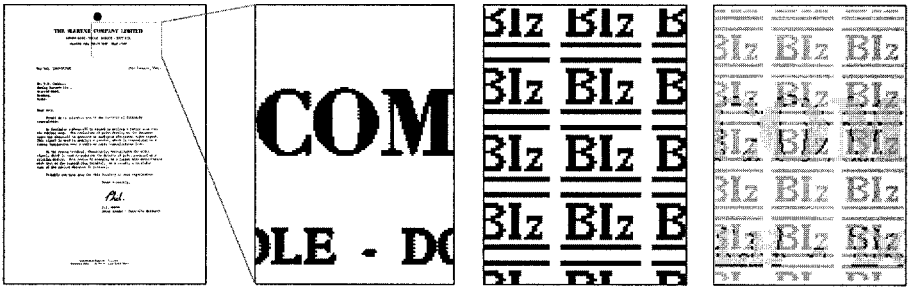
where $Q=2,4,6,...$ is a parameter which corresponds to robustness. The entire composite quantization table T^1 can then be written as:

$$T^1 = \left\{t = kQ + \bar{b}\frac{Q}{2} \mid b \in (0,1), k = 1,2,3...\right\}. \tag{23}$$

13.7 BLOCK PIXEL STATISTICS MANIPULATION

Another watermarking scheme embeds data bits into a binary document image by enforcing certain statistics of the black pixels in image partitioning blocks. This watermarking scheme is called the Block Pixel Statistics Manipulation (BPSM) watermarking scheme, and it is similar to the algorithm for digital watermarking of binary images presented in [26].

The BPSM watermarking scheme uses a binary logo pattern as a watermark. In order to embed this logo pattern into a binary document image, the image is partitioned into blocks of the logo pattern size. The watermark embedding process starts by selecting an initial set of pixel candidates for modification. This set is created using dilatation (expansion) of objects in the binary document image based on the vertical and horizontal neighbors of image pixels. The initial set of modification candidates contains the white pixels from the original document image that have either a horizontal or vertical black pixel neighbor. The watermark logo is embedded into the document image by filtering the initial set of modification candidate pixels according to the following rules, as illustrated by images of Figure 13.10:



A B C D

Figure 13.10. The BPSM watermark Embedding Process: A/ The original binary document image; B/ Zoomed in on a marked segment of the original image; C/ Zoomed in on one segment of an image created by periodic repetition, i.e. tiling, of the watermark logo pattern; D/ Overlapped images B/ and C/ to demonstrate filtering of the set of pixel candidates for modification: Black pixels from B/ and C/ are shown as different shades of gray, and pixels from the filtered set of modification candidates are shown in black.

Let W be the watermark logo pattern, and let C be the original binary document image. The T_W image is created by tiling the watermark logo across the size of the original image C. An example of one section of T_W is shown in Figure 13.10.C. This image tiling operation represents the partitioning of image C into the watermark embedding blocks. Let MC be the initial set of modification candidate pixels. The filtered set of modification candidate pixels, MC_F, is the subset of MC, $MC_F \subseteq MC$, defined by:

$$C(i,j) \in MC_F \quad iff \quad C(i,j) \in MC \wedge T_W(i,j) \; is \; black. \tag{24}$$

In other words, the MC_F represents the set of pixels created by a dilatation of C filtered by T_W, and the watermarked image is given by $C_W = C \cup MC_F$.

Given a test image C_T, a watermark is detected by partitioning the test image into the watermark detection blocks, and taking the sum of all pixels in each individual watermark detection block. Watermark detection blocks are created by taking all image pixels at the same relative position in the watermark embedding blocks created earlier and then combining them into individual blocks. This technique will work only if white pixels are encoded as zeros, and black pixels as ones. If the encoding of black and white pixels is

opposite to this, such as the encoding used by the MathWorks MATLAB, the images have to be complemented first. The watermark detection process can be described more formally as:

$$B_{i,j} = \left\{ p_{i+w_x*k_x, j+w_y*k_y} \in C_T \mid k_x, k_y = 0,1,...; \right\} \tag{25}$$

$$\tilde{w}_{i,j} = \sum_{k_x,k_y} B_{i,j} \tag{26}$$

$$\tilde{W} = \left\{ \tilde{w}_{i,j} \mid i = 1..w_x, j = 1..w_y \right\} \tag{27}$$

$$W_r = erf\left(\frac{\tilde{W} - E(\tilde{W})}{std(\tilde{W})} \right), \tag{28}$$

where *erf* is the error function used to enhance the dynamic of the extracted watermark image calculated in (12).

Watermark detection is possible because watermark embedding has changed the statistics of the black pixels in the watermarked image, so that black pixels of the watermarked image have higher probability of being black pixels of *Tw*. Figure 13.11 shows recovered marks from eight CCITT images given in Figure 13.5.

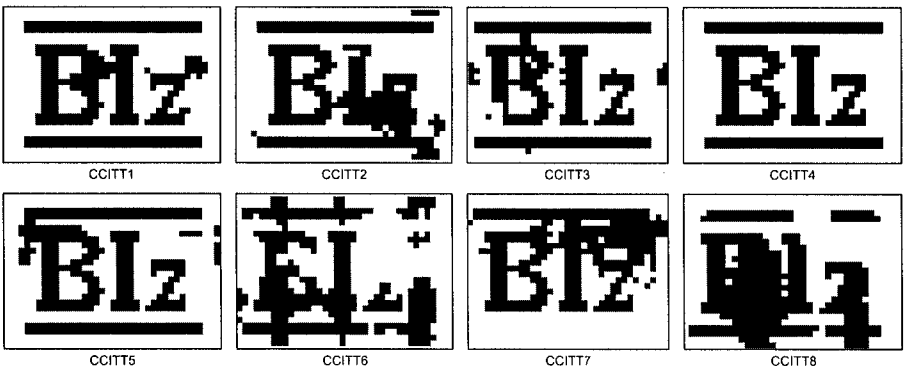


Figure 13.11. The Biz watermark logo recovered from the set of CCITT binary images.

Performance of this watermarking scheme depends on both the percentage of modification candidate pixels in an image, and the distribution of modification candidate pixels across an image. For example, the CCITT4 image, a text document image with many

evenly distributed symbols of French alphabet, is very suitable for BPSM watermark embedding because it has a high percentage of modification candidates (2%), and those modification candidates are evenly spread across the image. The number of modification candidate pixels is important because it ensures that statistics of black pixels in the watermarked document will be sufficiently modified to allow detection of the embedded watermark. Uniform distribution of those candidate pixels is important because it ensures that the statistics of each individual black watermark pixel will be changed sufficiently to minimize detection error. Figure 13.11 supports that argument because it shows that the embedded watermark has been extracted from the CCITT4 document image with no errors. On the other hand, some images not only have a small percentage of modification candidate pixels, but also have those modification candidate pixels localized in certain regions of the image. The CCITT6 and CCITT8 document images are representatives of those types of images. They have 0.7% and 0.6% of modification candidate pixels respectively, and those pixels are not uniformly distributed. Those two images are not good candidates for BPSM watermark embedding because the extracted watermarks are expected to have many errors. Figure 13.11 supports that analysis. It shows that two extracted watermarks from CCITT6 and CCITT8 images do not even resemble the embedded Biz logo image.

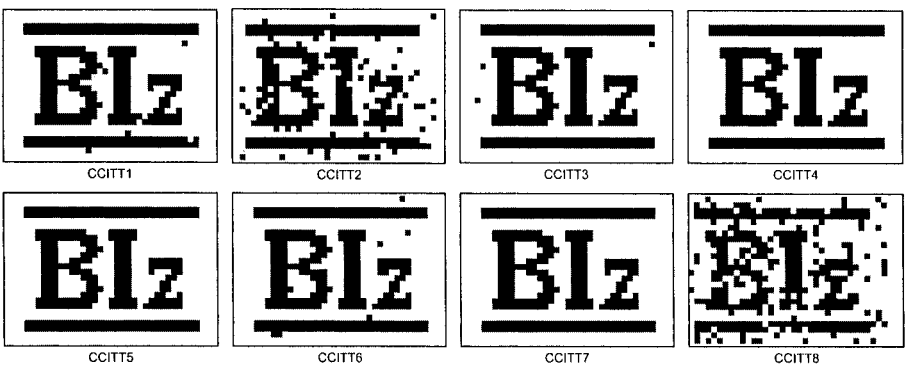


Figure 13.12. The Biz logo recovered from the set of CCITT images after enhancing the original watermarking scheme by ensuring that modification candidate pixels are evenly distributed.

Distribution of modification candidate pixels can be made more even by performing a random permutation of the candidate pixels before

using them to embed a watermark. The implementation of that enhancement substantially improves the watermark detection performance, as demonstrated by Figure 13.12.

The quality of the extracted logo image has improved, which means that the watermark detection error rate is lower. Even in a few cases where the extracted watermark has a number of errors, e.g. CCITT2 and CCITT8, the errors did not corrupt the extracted logo image beyond recognition.

13.8 MULTILEVEL WATERMARKS

Different applications use watermarks for different purposes. Watermarks can be used for copyright protection, owner identification, authentication, and tamper detection, to name a few applications. Each individual application has its own set of mutually-conflicting requirements for data embedding capacity, robustness, and quality. Two or more watermarks can be embedded into a cover signal (e.g. binary document image) to simultaneously satisfy the requirements of multiple applications.

Multilevel watermarks can be embedded in non-overlapping areas of the cover signal. This can be done by dividing the information carrying coefficients of the cover signal into multiple groups and embedding different watermarks into different groups of coefficients. This division can be done in time, space or frequency. An example of a two-level watermarking system, which is based on partitioning of the spectrum in a block DCT domain, is presented in [78][79]. Low-band coefficients are used to embed a watermark with high-capacity and moderate robustness, and mid-band coefficients are used to embed a watermark with low-capacity, and high robustness. A scheme for embedding two watermarks simultaneously into non-overlapping sets of wavelet coefficients is presented in [49]. Multilevel watermarks are usually embedded into non-overlapping segments to avoid interference among different levels.

Multilevel watermarks can also be embedded into a cover signal in an overlapping fashion, where different watermarks are embedded successively, and they all share the same watermark embedding space. Overlapped embedding is susceptible to interference and conflicts among the different watermark levels. Unlike non-overlapped watermark embedding, overlapped embedding requires

that a strict order of embedding be followed. The watermarks are embedded in order of decreasing robustness, and the fragile watermark is embedded last. Embedding a robust watermark after a fragile one is not possible, because any modification of image after the fragile mark has been embedded will damage the fragile mark enough to make it undetectable. This is a consequence of fragile mark design, because by its design, a fragile mark should be damaged by any operation that alters the cover signal, and robust watermarking is definitely such an operation [54].

A unified mathematical framework for the simultaneous embedding of two watermarks is presented in [48]. Two-level watermarking is modeled as broadcast communication with side information known at the encoder. Overlapped embedding was shown to achieve higher data hiding capacity than non-overlapped embedding, and in the case of overlapped embedding of robust and fragile watermarks, optimal data hiding requires that the robust watermark is embedded first.

Some watermarking applications require a watermark to be spread across the entire cover signal. For example, if a watermark is used to detect document tampering, those parts of the document not covered with a watermark will not be protected, and any tampering in those areas will not be detectable. Because of that, multilevel watermarking applications, which use one watermark for authentication or detection of document tampering, have to embed watermarks in the overlapping fashion.

Chapter 14

TWO-LEVEL MARKS: IMPLEMENTATIONS

This chapter presents a number of experimental results that demonstrate the evolutional development of the adaptive two-level watermarking scheme suitable for binary document images. Experimentation started by comparing the results of the SNDM-based pixel scoring method with the MWLUT pixel scoring [80]. The purpose of image pixel scoring is to identify the best pixel candidates for modification. The best candidates are those pixels whose modification will result in minimum visually perceptible image distortion. The results presented in this chapter show that both pixel scoring methods identify modification candidate pixels whose modification cause small visually perceptible image distortion. However, when watermark embedding modifies a large number of image pixels, the SNDM-based scoring identifies the better set of modification candidate pixels than MWLUT scoring. The complexity of the algorithm which calculates the SNDM pixel scores in a 3×3 pixel neighborhood is the same as the complexity of algorithm which calculates the lookup table access key based on a 3×3 pixel neighborhood. Additionally, unlike the MWLUT method, which only uses pixel's 3×3 neighborhood for scoring, the SNDM-based pixel scoring method can be easily extended to larger pixel neighborhoods, such as 5×5 and 7×7.

The SNDM-based pixel scoring is used as an important component of all two-level watermarking schemes presented in this document. The first watermarking scheme embeds two watermarks into a binary document image using overlapped embedding. One watermark carries document ownership information, and it is embedded as a robust mark using the Block Pixel Statistics Manipulation (BPSM) watermarking method. The other watermark is used for document authentication, and it is embedded using a watermarking scheme based on the Uniform Quantization with the quantization step $Q=2$. This two-level watermarking scheme creates a lot of visible image distortion, but watermark detection works

correctly, and it successfully detects both embedded watermarks in all eight test images. Additionally, image tampering was successfully detected in all cases. Visible image distortion is caused by the BPSM watermarking component, which modifies a large number of image pixels. This visible distortion of watermarked images can be reduced by reducing the number of pixels the BPSM modifies to embed watermark. The number of pixels to modify can be reduced by identifying the pixels whose modification will introduce visible distortion and removing those pixels from the set of modification candidate pixels. The selection of pixels to remove from the set of modification candidate pixels is based on SNDM scores. The modified two-level watermarking scheme, which controls the number of pixels the BPSM watermarking component modifies, helps reduce visible distortion in all watermarked images. However, it also increases watermark detection error.

Since the BPSM watermarking scheme either creates visible image distortion or has high watermark detection error, it is not suitable for the robust watermark embedding, and it is replaced with the Uniform Quantization watermarking method. The resulting new two-level watermarking scheme uses uniform quantization for both robust and fragile embedding. The robust watermark embedding uses quantization step $Q=10$, and the fragile mark embedding uses quantization step $Q=2$. This second watermarking technique represents an improvement over the previous watermarking scheme because it keeps the same level of watermark detection performance, but it substantially reduces visible distortion in the watermarked images.

The third watermarking method, called the adaptive two-level watermarking scheme further reduces visible distortion of watermarked images. For a given binary document image, the adaptive watermarking scheme selects the image partitioning block size (e.g. 16×16, 32×32 or 64×64), the version of random image permutation (e.g. permutation based on the key1, key2 or key3), and the size of pixel neighborhood where pixel modification score is calculated (e.g. 3×3, 5×5, 7×7) to minimize visible distortion of watermarked image. Visible image distortion is measured using the Distance Reciprocal Distortion Measure (DRDM).

14.1 EVALUATION OF SNDM PIXEL SCORING

The objective of this experiment is to evaluate performance of the SNDM pixel scoring method. This is accomplished by embedding watermarks into the set of CCITT images given in Figure 13.5. Watermarks are embedded by modifying some subset of image pixels, where pixels are selected for modification based on two different pixel scoring methods, the SNDM and MWLUT. Then the resulting image distortion is calculated using two different distortion metrics, a Peak Signal-To-Noise Ratio (PSNR), and a Distance-Reciprocal Distortion Measure (DRDM) [50]. This experiment is illustrated in Figure 14.1.

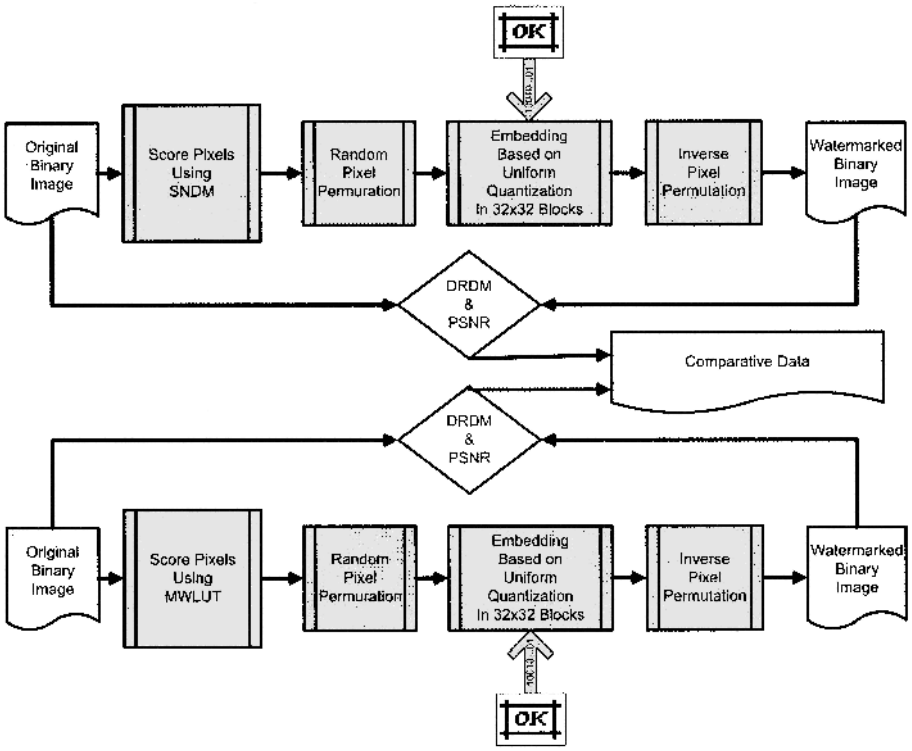


Figure 14.1. The two watermark embedding schemes differ only in how the modification candidate pixels are scored. One scheme uses the SNDM scoring and the other one the MWLUT scoring.

Data embedding is based on partitioning an image into 64×64 blocks and hiding one bit per block using the uniform quantization

approach. The size of all CCITT test images is 2376×1728, so that with 64×64 partitioning blocks, the embedding capacity of this watermarking scheme is 999 bits. Two 910-bit logo images, the Biz and OK, given in Figure 13.1, are embedded as watermarks. Image pixels are randomly permuted to ensure a more even distribution of modification candidate pixels. The candidate pixels to be modified are selected based on the SNDM scores calculated in the 3×3 pixel neighborhood in one case, and based on the MWLUT scores stored in a pre-calculated lookup table of 512 entries in the other case. The MWLUT lookup table contains the flipping scores for all possible 3×3 patterns. The overall image distortion caused by watermark embedding is measured using both PSNR and DRDM. The DRDM uses 7×7 weight matrix W_7.

Table 14.1 provides a summary of image distortion results calculated using PSNR and DRDM after embedding the OK logo as a fragile mark into the set of CCITT images. The OK logo image is embedded using the uniform quantization with the quantization step $Q=2$. The PSNR numbers calculated for the two watermarking schemes using the two pixel scoring methods are either the same or very close to each other. Since PSNR provides a measure of the number of modified pixels in a binary image, both watermark embedding methods modify approximately the same number of pixels in order to embed the 910-bit OK logo message. The number of pixels, which are modified to embed the OK logo as a fragile mark into the set of CCITT images, is provided in Table 14.2. The table shows that the number of modified pixels is approximately equal to a one half of the number of message bits. In other words, in order to embed a 910-bit message, approximately one half of the image partitioning blocks do not need to be modified at all.

The PSNR numbers demonstrate that both watermark embedding schemes modify approximately the same number of image pixels. However, those numbers do not give any indication about whether image pixel modifications create visible image distortions. The DRDM represents a measure of visually perceptible image distortion. The DRDM numbers listed in Table 14.1 are very small. The values indicate very small visually perceptible distortion of watermarked documents, and consequently a very good quality of watermarked document images. In general, watermarked images with DRDM values below 0.2 have a good quality and very little visible distortion

[50]. Accordingly, the watermarked images with DRDM numbers close to zero have no visible distortion.

Table 14.1. Comparison of image distortion results measured using PSNR and DRDM between watermark embedding schemes based on the MWLUT pixel scoring and the SNDM pixel scoring. In both cases, the 910-bit OK logo message was embedded into the set of CCITT images as a fragile mark.

	1	2	3	4	5	6	7	8
PSNR (SNDM)	27.03	26.69	26.88	26.92	26.91	26.98	26.89	27.06
PSNR (MWLUT)	27.06	26.91	26.88	26.92	26.91	27.02	26.89	27.06
DRDM (SNDM)	0.006	0.009	0.007	0.001	0.004	0.009	0.011	0.003
DRDM (MWLUT)	0.029	0.036	0.003	0.001	0.004	0.016	0.004	0.010

An attempt to demonstrate that watermarked images have no visible artifacts can be found in Figure 14.2 where the original CCITT1 image is shown side by side with its watermarked version where pixels were modified based on their SNDM and MWLUT scores. Both pixel scoring methods obviously identified good modification candidates, so that a naked eye cannot see the difference between the original and the watermarked image.

Table 14.2. The number of pixels modified as part of embedding the OK logo as a fragile mark.

	1	2	3	4	5	6	7	8
MSE (SNDM)	505	467	488	492	491	499	489	508
MSE (MWLUT)	508	491	488	492	491	503	489	508

A subset of modified pixels is shown in Figure 14.3. The watermarked images are represented as gray scale images to make it easier to see which pixels have been modified as part of the watermark embedding process. Circles were used to mark the modification areas, and the 5×5 neighborhood of the modified pixels was marked with the lighter shade of gray. The modified pixels are either shown as black or white dots.

This experiment shows that embedding a 910-bit message as a fragile mark into the set CCITT images, where image pixels are modified based on their SNDM score, creates no visible image distortion, and the resulting quality of the watermarked image is comparable with the quality of the watermarked image where image pixels are modified based on their MWLUT scores.

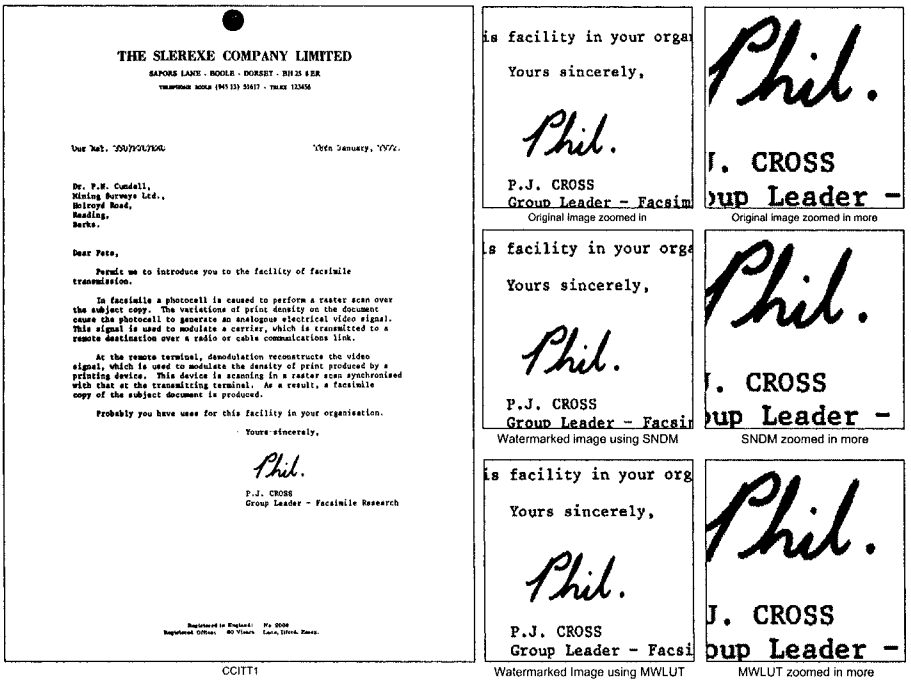


Figure 14.2. CCITT1 document is shown before and after watermark embedding. Document image pixels are selected for modification based on their SNDM scores.

Since embedding a 910-bit message as a fragile mark into the set of CCITT images results in modification of a very small fraction of image pixels (e.g. 508 out of 2376×1728, or 0.012% in the CCITT1 case), the next step was to evaluate visible image distortion caused by watermark embedding which results in modification of a larger number of pixels. To achieve that goal, this experiment was repeated with the 910-bit Biz logo message embedded as a robust watermark using the uniform quantization with the quantization step $Q=10$.

Even after embedding the 910-bit Biz logo message with the quantization step $Q=10$, the calculated DRDM results demonstrated that both pixel scoring methods, the SNDM and MWLUT, select pixels for modifications which do not create visible image distortions. Obviously, it is important to find a way to increase the number of modified pixels, because as the number of pixel modifications increases the resulting image distortion should become more visible.

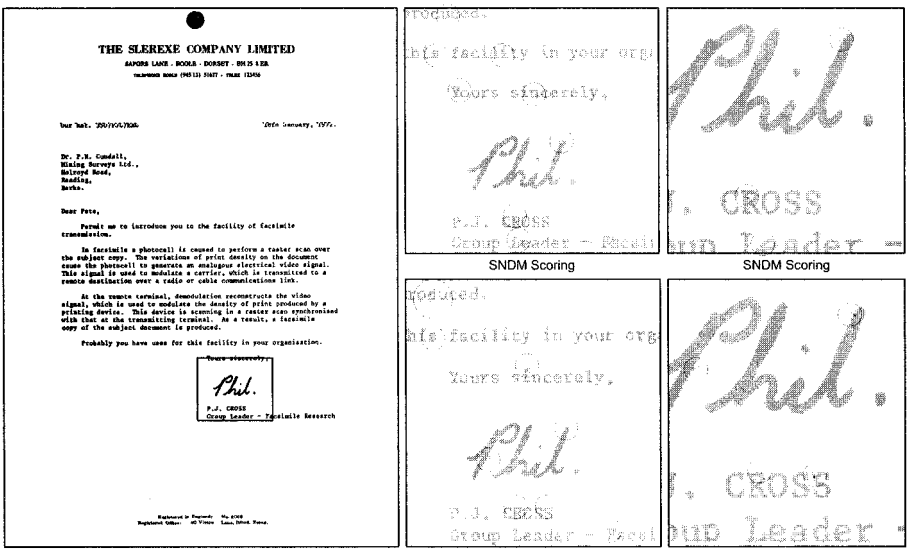


Figure 14.3. Watermarked versions of an enlarged segment of the CCITT1 image. For presentation purposes only, this binary image was turned into a gray scale image to help spot the modified pixels. The modified pixels are marked black or white dots, and their 5x5 neighborhood is painted with the lighter shade of gray. Those areas are also marked with circles.

The number of pixel modifications can be increased by making the embedded Biz logo more robust, and this is achieved by increasing the quantization step *Q*. The next experiment was designed to achieve that objective. It is very similar to the previous one. The only difference is that this time, the 910-bit Biz logo message is embedded more robustly with the quantization step *Q=22*. More robust embedding will require more image pixels to be modified, so that two different pixel scoring methods should result in different levels of visible distortion in watermarked images.

Table 14.3 provides image distortion information measured using MSE and DRDM for watermark embedding schemes based on MWLUT and SNDM pixels scoring using uniform quantization with the quantization step *Q=22*. The MSE numbers indicate that document images CCITT4 and CCITT7 have the same numbers of pixels modified. This means that both pixel scoring methods have identified enough modification candidate pixels to embed the Biz logo message robustly into CCITT4 and CCITT7. The DRDM measure is at least 5 times larger when embedding is done using the MWLUT scoring, indicating that the Biz embedding will create less visible distortion when pixels are modified based on their SNDM scores.

Table 14.3. Comparison of image distortion results measured using MSE and DRDM between watermark embedding schemes based on MWLUT pixel scoring and SNDM pixel scoring. In both cases the 910-bit Biz logo massage was embedded into the set of CCITT images as a robust watermark with Q=22.

	CCITT4	**CCITT7**
MSE (SNDM)	4524	4513
MSE (MWLUT)	4524	4513
DRDM (SNDM)	0.0023	0.0066
DRDM (MWLUT)	0.0096	0.0206

Figure 14.4 confirms that embedding based on the SNDM pixel scoring results in watermarked images with less visible distortion than embedding based on the MWLUT scores. The Biz logo message is embedded as a robust watermark into the CCITT7 document

image using uniform quantization with step size Q=22. Visual inspection of the zoomed in portion of the watermarked document image reveals that various artifacts are more visible when watermark is embedded using pixel candidates selected according to their MWLUT scores. This visual inspection of watermarked images supports DRDM visible distortion results presented in Table 14.3.

Figure 14.4. The result of embedding the Biz logo as a robust message using uniform quantization with step size Q=22. Image pixels are modified based on their SNDM score in one case and based on the MWLUT in the other case. The DRDM measure suggests that distortion is more visible when embedding is done based on the MWLUT scores. For example, look closely to the R part of the zoomed in portion of the watermarked document. The embedding based on SNDM scoring appears to be more compact.

In conclusion, both pixel scoring methods, the SNDM and the MWLUT, identify a set of modification candidate pixels whose modification causes little visible image distortion. When watermark embedding requires a large number of image pixels to be modified, the SNDM-based scoring identifies the better set of modification

candidate pixels than MWLUT scoring. Consequently, watermark embedding based on the SNDM pixels scoring creates fewer artifacts and results in smaller visible distortion of watermarked images. Additionally, the SNDM scoring has an advantage over the MWLUT scoring because it does not depend on a pre-calculated lookup table. It is computationally simple enough so that the lookup table is not necessary, and the pixel modification scores can be calculated for each pixel when and as needed. This also means that the SNDM based scoring can be easily extended to larger neighborhoods, such as 5×5 or 7×7, unlike any LUT-based scoring including the MWLUT, which would require very large lookup table with 32,768K entries in order to support scoring pixels based on their 5×5 neighborhoods, for example.

Actually, the computational complexity of calculating the SNDM score is equivalent to the computational complexity of calculating the lookup table access key based on a pixel neighborhood. For example, the SNDM score for an image pixel in the 3×3 neighborhood is calculated by convolving the 3×3 image pixels with the 3×3 weight matrix, as follows:

$$SNDM_{p_{i,j}} = \sum \left(\begin{bmatrix} p_{i-1,j-1} & p_{i-1,j} & p_{i-1,j+1} \\ p_{i,j-1} & p_{i,j} & p_{i,j+1} \\ p_{i+1,j-1} & p_{i-1,j} & p_{i+1,j+1} \end{bmatrix} \circ \begin{bmatrix} 0.1029 & 0.1471 & 0.1029 \\ 0.1471 & 0 & 0.1471 \\ 0.1029 & 0.1471 & 0.1029 \end{bmatrix} \right), \quad (29)$$

where $p_{i,j}$ is an image pixel an SNDM score is calculated for, and $\circ$ is an element-wise matrix multiplication operation. The lookup table access key in the same pixel neighborhood is calculated as follows:

$$LUT\_key_{p_{i,j}} = \sum \left(\begin{bmatrix} p_{i-1,j-1} & p_{i-1,j} & p_{i-1,j+1} \\ p_{i,j-1} & p_{i,j} & p_{i,j+1} \\ p_{i+1,j-1} & p_{i-1,j} & p_{i+1,j+1} \end{bmatrix} \circ \begin{bmatrix} 256 & 32 & 4 \\ 128 & 16 & 2 \\ 64 & 8 & 1 \end{bmatrix} \right), \quad (30)$$

In order to read pixel modification scores from the pre-compiled lookup table, the calculation given in the Equation (15) is performed. It is obvious that the element-wise multiplication of two 3×3 matrices followed by summation of all matrix elements required for calculation of the LUT access key is equivalent to the element-wise multiplication of two matrices followed by summation of all matrix elements performed by Equation (14), which calculates the SNDM score. Therefore, the computational complexity of calculation of

SNDM score is equivalent to the computational complexity of accessing a pre-compiled score from the LUT.

14.2 BPSM AND UNIFORM QUANTIZATION

The objective of this experiment is to evaluate the performance of one specific two-level watermarking scheme suitable for binary document images. This watermarking scheme embeds two watermarks into the original binary document image in an overlapping fashion. One watermark conveys side information, e.g. document ownership, and it is embedded as a robust mark using the BPSM watermarking scheme. The other watermark is used for document authentication and tamper detection, and it is embedded using a watermarking scheme based on uniform quantization with quantization step $Q=2$, and 64×64 image partitioning block size. The Biz logo binary image is used as a document ownership message, and the OK logo binary image is used as an authentication message. If a modification is made to any part of the watermarked binary document image, the OK logo authentication message will no longer be detectible indicating that the document was tampered with. This document tampering does not destroy the ownership message, which can still be detected and extracted. This experiment is illustrated in Figure 14.5.

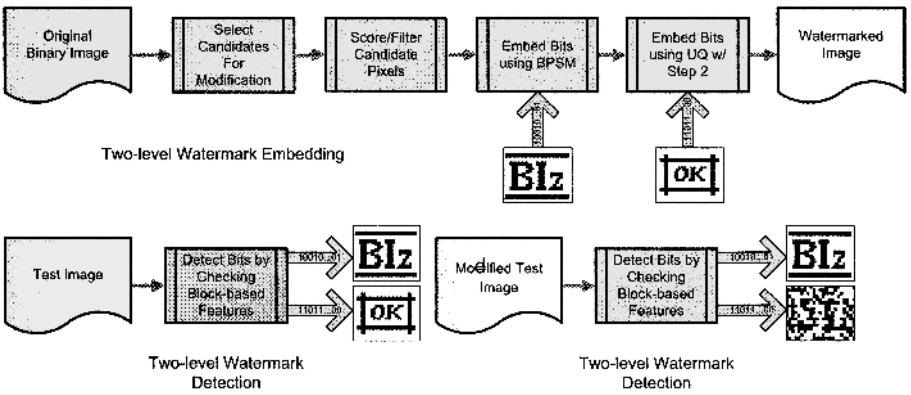


Figure 14.5. Block diagram of the two-level watermarking scheme with overlapped embedding. The first message is embedded as a robust watermark using the BPSM watermarking method and the second message is embedded as a fragile mark using the Uniform Quantization method with the quantization step $Q=2$. Both watermarks can be detected by checking block-based features in a

watermarked test image. However, if the watermarked image is tampered with, the fragile mark will not be detected, indicating that this test image is not an authentic image, but the robust mark will still be detectable.

This watermarking scheme was applied to the set of eight CCITT binary document images, and one result is illustrated in Figure 14.6, which shows the original CCITT5 image, together with the zoomed in segment of the same section in the original and watermarked image. The figure also shows two watermarks extracted from the watermarked image which was not tampered with. A general observation to be made is that watermarked images have much of visible distortion, which can be seen in all zoomed in segments of all watermarked CCITT images. On the positive side, watermark detection performs well and all extracted watermarks are recognizable.

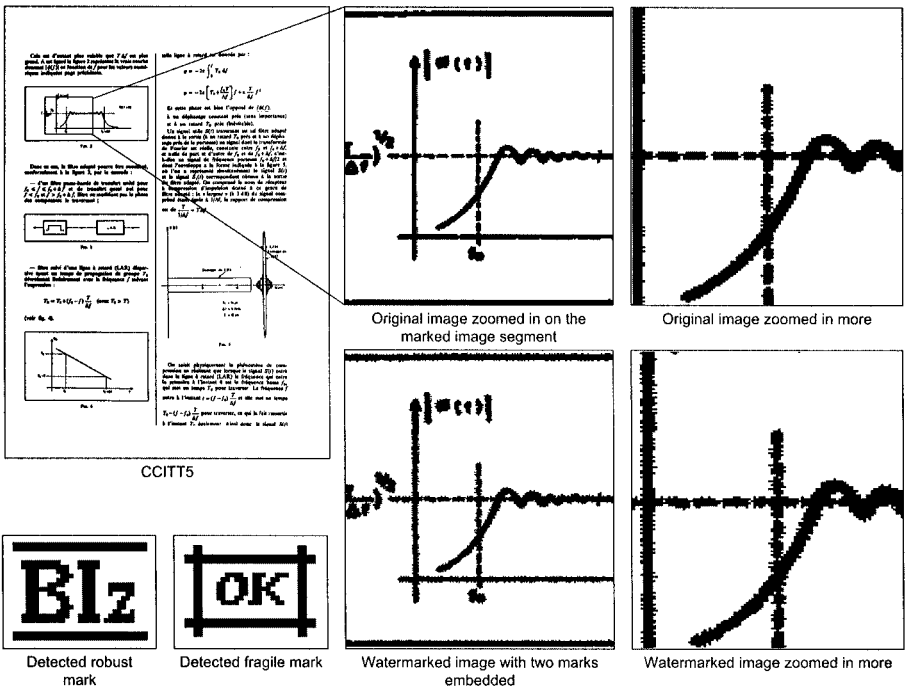


Figure 14.6. The result of embedding Biz and OK marks into the CCITT5 binary document image

The measure of distortion caused by embedding two watermarks, Biz and OK, into the set of CCITT images is given in Table 14.4.

In order to examine the impact image tampering has on watermark detection abilities of the two-level watermarking scheme, it is useful to be able to asses how many image pixels need to be modified to cause a significant document change. A significant change may be a change of one character in a document, a change that for instance, turns $1000 into $9000. The number of pixels used to create a single character depends on the size of the font. For the test documents, it was determined that approximately 128 pixels in a binary document image would have to be modified to change one document character, as depicted in Figure 14.7.

Table 14.4. Measure of distortion of the set of CCITT watermarked images. Two watermarks have been embedded into each CCITT image. The Biz logo message was embedded first as a robust watermark using the BPSM watermarking method, and the OK logo message is embedded after that as a fragile mark using the Uniform Quantization method with the quantization step Q=2.

	1	2	3	4	5	6	7	8
MSE	23809	13529	43040	94646	45776	28090	85603	25230
PSNR	43.76	41.31	46.33	49.76	46.60	44.48	49.32	44.02
DRDM	4.837	2.528	4.719	8.947	5.616	3.737	9.883	2.57

Based on the assumption that a large number of document pixels have to be modified to create a significant document content change, an experiment was devised where 0, 64, 128, 192, and 256 pixels in the set of watermarked CCITT images were modified. The impact those modifications had on the ability of the watermarking scheme to extract the Biz and OK watermarks was then observed. The results of pixel modifications applied to the document image CCITT5 is shown in Figure 14.8, where it can be seen that after 128 image pixels are modified, the extracted OK logo image, the image embedded as a fragile mark, is no longer recognizable. At the same time, the extracted Biz logo image, the image embedded as a robust mark, is unaffected by the modifications of the watermarked image.

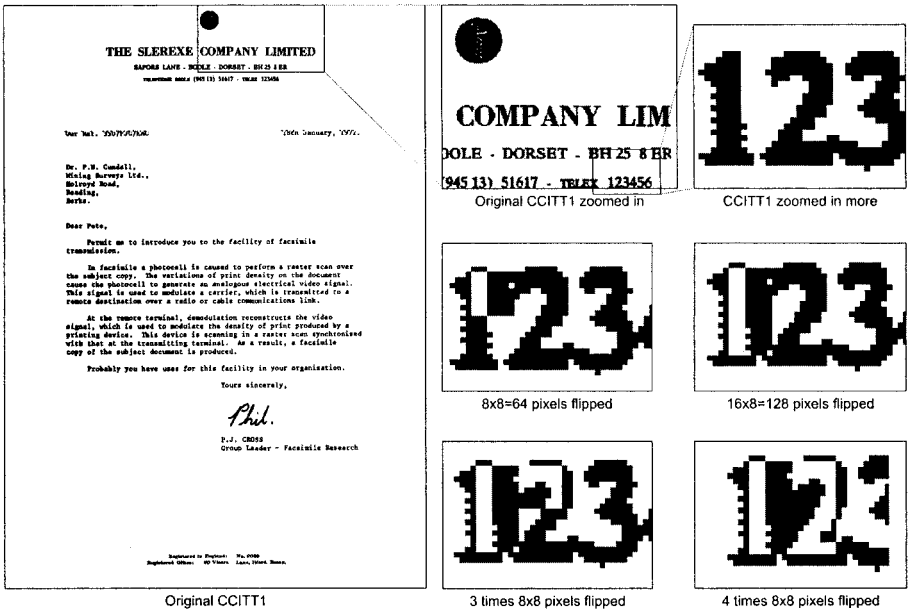


Figure 14.7 Results of modification of 64, 128, 192, and 256 pixels in an image. Approximately 128 pixels need to be modified in a binary document image to change one character.

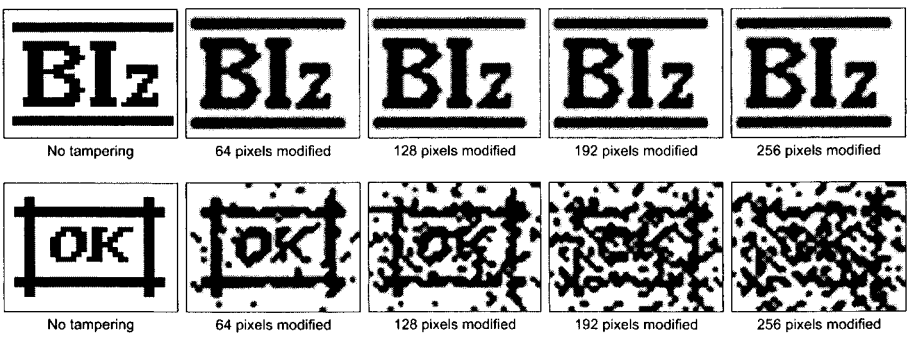


Figure 14.8 Two watermarks, Biz and OK extracted from CCITT5 binary document image after 0, 64,128,192 and 256 pixels have been modified.

In conclusion, this two-level watermarking scheme introduces visible image distortion. Both watermarks can be reliably extracted and detected, and image tampering which modifies more than 128 pixels of the watermarked image renders the fragile OK logo message

unrecognizable. The robust Biz logo image survives image tampering operation.

This watermarking scheme causes visible image distortion because it modifies a large fraction of image pixels while embedding the Biz logo as a robust watermark. For example, Table 14.4 shows that this watermarking scheme modified 45776 image pixels, which represents 1.11% of image pixels, to embed two 910-bit binary logo images Biz and OK into the CCITT5 document image. This watermarking scheme could be improved by modifying fewer image pixels as part of watermark embedding operation, provided that the watermark detector is still able to extract and detect the embedded watermark.

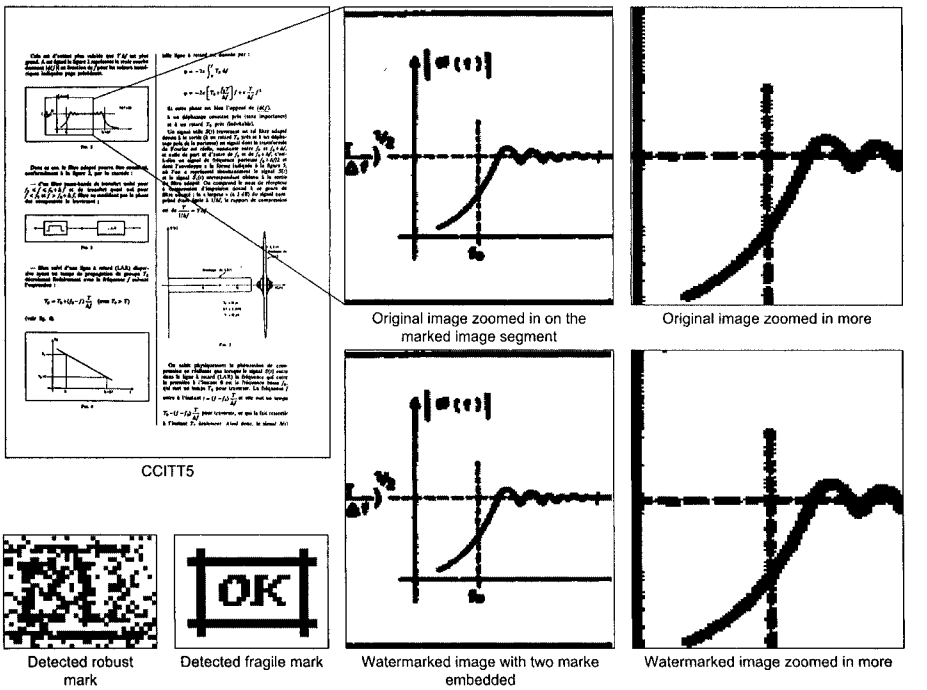


Figure 14.9. The result of embedding Biz and OK marks into the CCITT5 binary document image after the number of modified pixels as part of robust Biz mark embedding had been reduced in order to limit image distortion.

The original BPSM watermarking scheme created an initial set of modification candidates by taking into consideration all white 4-

neighbor pixels of black pixels. Those are all white horizontal and vertical neighbors of the black pixels in a binary document image. This original BPSM scheme was modified by pruning this initial set of modification candidates before using them for data embedding. The pruning was based on the SNDM scores calculated for all pixels in the initial set of modification candidates. In order to improve the quality of watermarked images, all pixels with low SNDM scores (SNDM < 0.35) were removed from the initial set of modification candidates. The results of modified BPSM experiment applied to the CCITT5 document image is illustrated in Figure 14.9.

Visual inspection of the zoomed in portion of the watermarked document image CCITT5 reveals that the quality of the watermarked image has improved. Table 14.5 confirms the improvement and shows that the DRDM quality measure is much lower. It also shows that a much smaller fraction of document image pixels needed to be modified in order to embed two watermarks into the CCITT document images. Unfortunately, this modified watermarking scheme could no longer detect the robust watermark.

Table 14.5. Measure of distortion of the set of CCITT watermarked images. Two watermarks have been embedded into each CCITT image. The Biz logo message was embedded first as a robust watermark using the BPSM watermarking method, and the OK logo message is embedded after that as a fragile mark using the Uniform Quantization method with the quantization step Q=2.

	1	2	3	4	5	6	7	8
MSE	5357	3098	8372	23091	9699	4628	21003	5863
PSNR	37.29	34.91	39.22	43.63	39.86	36.65	43.22	37.68
DRDM	0.167	0.087	0.154	0.391	0.192	0.104	0.498	0.093

14.3 UNIFORM QUANTIZATION APPLIED TWICE

It has been shown that the watermark detection performance of the previous two-level watermarking scheme depended on the watermark detection performance of the BPSM watermarking scheme, the watermarking component that was used for robust embedding. Since the original BPSM watermarking scheme had to

modify a large fraction of image pixels as part of watermark embedding operation, the resulting watermarked images had a lot of visible artifacts. Reducing the number of pixel modifications did not work. It did produce better quality watermarked images, but this modified BPSM watermarking scheme could not detect watermarks any longer.

The objective of this experiment is to replace the BPSM watermarking scheme with the scheme based on Uniform Quantization with quantization step Q=10, as illustrated by Figure 14.10, and verify how much this change will affect the quality of watermarked images and the watermark detection performance.

The results of this experiment applied to the CCITT4 document image is illustrated in Figure 14.11 which demonstrates that the quality of watermarked image is very high, and that the embedded marks can be extracted and successfully detected. The high quality of all watermarked images is confirmed by performance statistics provided in Table 14.6.

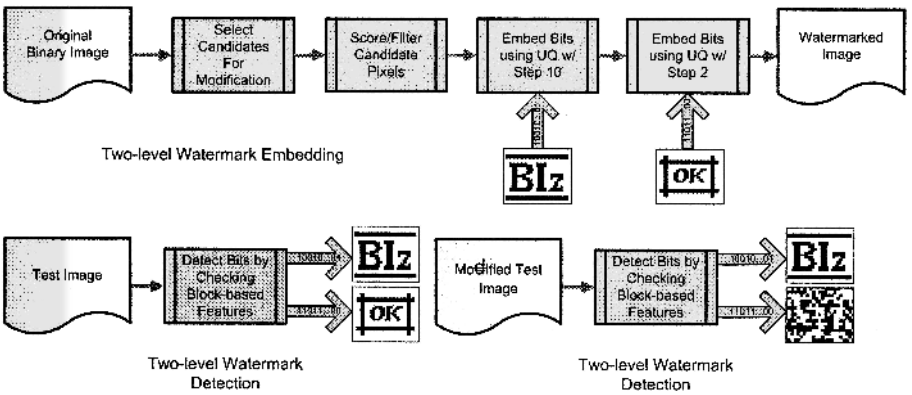


Figure 14.10 Block diagram of two-level watermarking scheme with overlapped embedding. The first message is embedded as a robust watermark using the uniform quantization method with quantization step Q=10, and the second message is embedded as a fragile mark using the uniform quantization method with Q=2.

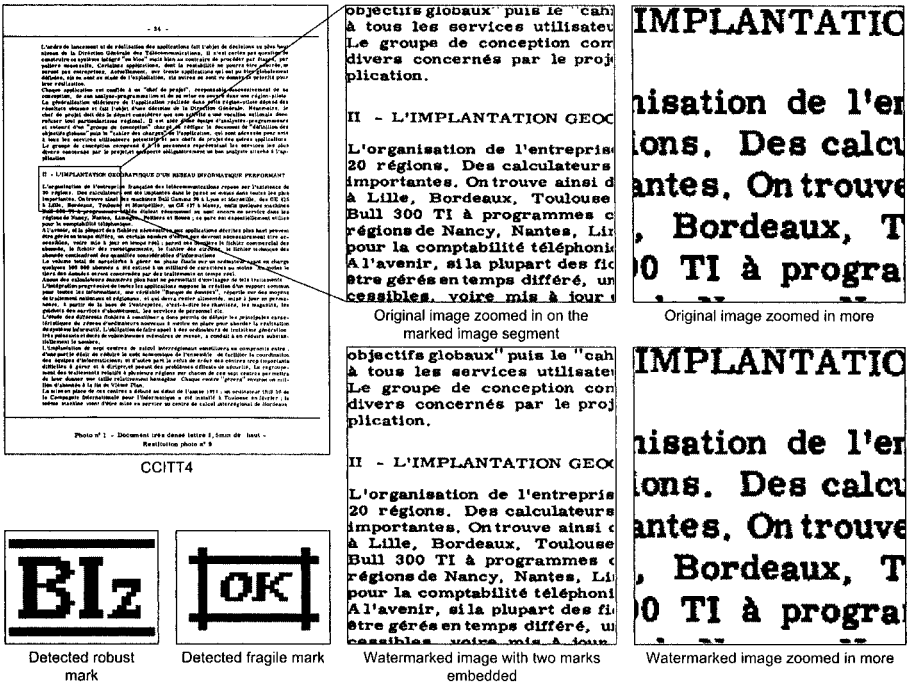


Figure 14.11 Result of embedding Biz and OK marks into the CCITT4 binary document image.

Compared to the previous watermark embedding scheme, this one modifies significantly fewer pixels to embed two 910-bit logo images. For example, the previous method modified 94,646 pixels (2.3%) to embed the two logo images into the CCITT4 binary document image, and this improved method modified only 2312 pixels (0.048%) to achieve the same result. The DRDM numbers indicate that the modified pixels are selected properly so as not to create visible artifacts in the watermarked images. Figure 14.12 illustrates that this new two-level watermarking scheme is still sensitive to image tampering. Modification of a single document character worth of pixels renders the fragile mark undetectable, but it does not prevent the watermark detector from being able to detect the robust mark.

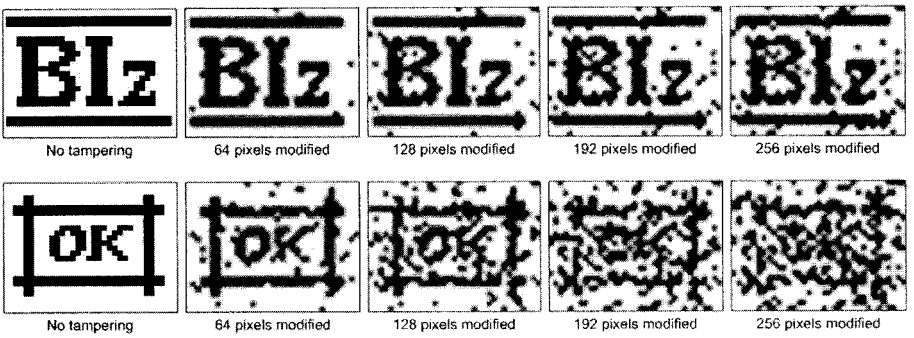


Figure 14.12 Two watermarks, Biz and OK, extracted from the binary document image CCITT4 after 0, 64, 128, 192 and 256 pixels have been modified.

Table 14.6. Measure of distortion of the set of CCITT watermarked images. Two watermarks have been embedded into each CCITT image. The Biz logo message was embedded first as a robust watermark using the Uniform Quantization watermarking method with quantization step Q=10, and the OK logo message is embedded after that as a fragile mark using the Uniform Quantization method with quantization step Q=2.

	1	2	3	4	5	6	7	8
MSE	2137	1849	2210	2312	2148	1995	2183	2140
PSNR	33.3	32.67	33.44	33.64	33.32	32.1	33.39	33.30
DRDM	0.013	0.012	0.004	0.002	0.005	0.010	0.007	0.004

14.4 ADAPTIVE TWO-LEVEL WATERMARKING

The two-level watermarking scheme based on uniform quantization used for both robust and fragile embedding shows good results, as demonstrated in the previous section. The robust embedding was based on the uniform quantization with the step size of 10. That means the embedder will have to modify at most 5 pixels per block to embed a single message bit. The embedded bit is robust because three or more pixels in a block need to be modified to cause the bit detection error. The embedding can be made more robust by increasing the quantization step size. This increase however will negatively affect the quality of the watermarked image because the

number of pixels that will have to be modified as part of this robust embedding will increase as well. For example, to design a robust embedding scheme which can survive modification of n block pixels, uniform quantization with the step size $Q=2(2n+1)$ should be used. This embedding scheme will require modifying at most $2n+1$ pixels per block.

Embedding an additional fragile bit into the block potentially requires modifying another block pixel. Therefore, the two-level watermarking scheme based on uniform quantization, which embeds robust and fragile watermarks into a digital binary image, where the robust mark has to survive modification of n pixels per block, may need to modify $2n+2$ pixels per block.

It is possible to optimize this two-level watermarking scheme by reducing the maximum number of modifications in each block by one. This optimization is based on an observation that adding the fragile bits will require changing ½ of a pixel per block on average because, on average, 50% of all the blocks will require a pixel change. After embedding the fragile bits, the robust bits will be slightly less robust in 50% of all the blocks because those blocks will be robust to changes of fewer bits. The optimized two-level watermarking scheme embeds a robust mark by modifying at most $2n$ pixels per block instead of $2n+1$. With that optimization, and before the fragile bits get embedded, all the robust bits will be less robust, because modification of fewer pixels per block will cause the embedded bit to be miss-detected. However, as the fragile bits get embedded, 50% of all the blocks will have an additional pixel modified, and those modifications will restore the original level of robustness to those blocks. Therefore, the optimized two-level watermarking scheme has the same level of robustness as the original scheme, but it modifies one fewer pixel per block, consequently producing watermarked images of better quality.

This two-level watermarking scheme will work well if all image partitioning blocks have at least $2n+1$ good modification candidate pixels, for a required robustness level n. An example of one successful case of two-level watermark embedding is illustrated in Figure 14.11. The original CCITT4 binary document image has a lot of modification candidate pixels with high modification scores, and those candidates are uniformly distributed across 64x64 partitioning blocks after performing permutation of image pixels. The result is a

good quality watermarked image, and the ability to accurately detect both robust and fragile marks.

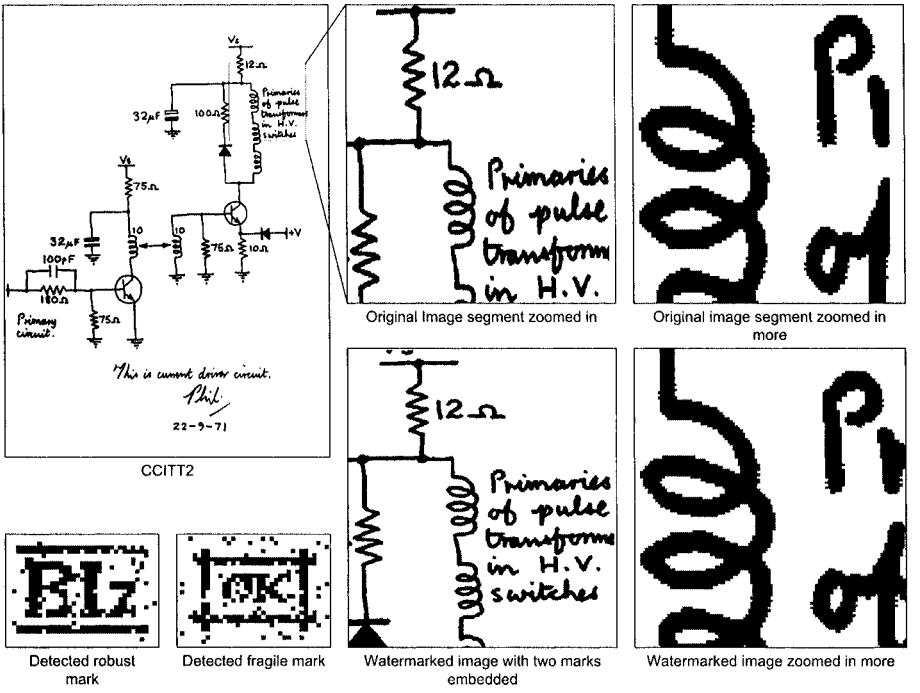


Figure 14.13 Result of embedding Biz and OK marks into the CCITT2 binary document image.

What happens if some partitioning blocks do not have a needed number of modification candidate pixels? In that case, no pixels will be modified, so the quality of the watermarked image will still be high, but the detector will not be able to detect those bits accurately. This is best illustrated in Figure 14.13 which shows the results of two-level watermark embedding and detection with CCITT2, the binary document image which does not have enough modification candidate pixels. The quality of the watermarked image is clearly high, but both the extracted marks have many errors.

The adaptive two-level watermarking scheme is illustrated in Figure 14.14. It has been designed to select the optimum embedding parameters which maximize embedding capacity, minimize visible

distortion of the watermarked image and improve watermark detection performance.

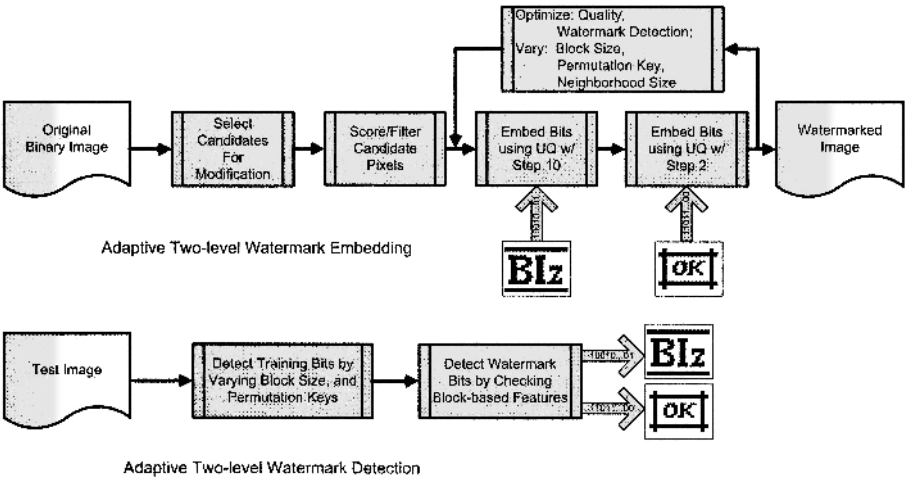


Figure 14.14 Adaptive two-level watermarking scheme: the embedder varies the block partitioning size, image permutation key, and the pixel scoring neighborhood size to optimize quality of the watermarked image and watermark detection performance.

This was achieved by varying the embedding parameters, partitioning block size, permutation key, and the size of pixel scoring neighborhood, and searching for the optimal combination. The optimal combination represents the minimum block partitioning size, which results in an acceptable visual quality of the watermarked image measured by the DRDM method, and a small watermark detection error measured by the MSE. Once the embedder identifies the optimum combination of embedding parameters, it uses those parameters to embed both watermarks. The watermarks are embedded starting from the 13^{th} bit position, after the self-verifying sequence of 12 training bits, which are used by the detector to identify the block size and the permutation key used by the embedder. The detector tries all the predefined combinations of the block size and permutation key until it finds a pair, which detects the training sequence. This pair is then used to detect the rest of the message.

Note that the detector does not need to identify the pixel scoring neighborhood used by the embedder, because this parameter does

not affect the detection process. The size of pixel scoring neighborhood is only used by the embedder to select the best set of modification candidate pixels.

The adaptive two-level watermark embedding algorithm can be described as follows:

1. Let B be a set of valid image partitioning block sizes B={b_i | $i=1,...m$};
   Let K be a set of valid image permutation keys (i.e the set of valid pseudo-random generator seeds), K = {k_j | $j=1,...n$};
   Let C be a binary image to be watermarked;
   Let W_f be a logo binary image to be inserted into C as a fragile mark;
   Let W_r be a logo binary image to be inserted into C as a robust mark;
   Let V be a 12 bit self-verifiable message to be inserted into C as the first 12 bits;
   Let N be a set of valid neighborhood sizes used for image pixel scoring.
2. Calculate maximum block size b from B which allows embedding of V+max(Wr,Wf) bits into the C;
   $b = \max(b_i) \therefore b \times \left(|V| + \max\left(|W_r|, |W_f|\right)\right) \le |C|$
3. Given image partitioning block size b, embed W_f and W_r into C using Standard two-level watermarking algorithm for each key k from K, obtaining $\tilde{C}_k, k = k_1,...k_n$; Embedding is done using all available modification candidate pixels, without making any attempts to minimize visible distortion of the watermarked images $\tilde{C}_k$
4. Use standard two-level watermarking algorithm to extract embedded watermarks, obtaining $W'_{r_k}, W'_{f_k}, k = k_1,...k_n$
5. Calculate watermark detection errors $E_{r_k} = MSE\left(W_{r_k}, W'_{r_k}\right)$, and $E_{f_k} = MSE\left(W_{f_k}, W'_{f_k}\right)$
6. Find k which optimizes watermark detection error as follows: $k' = \min_{k=k_1,...k_n}\left(E_{f_k}\right)$ $k = \min_{k'}\left(E_{r_{k'}}\right)$. In other words, minimize detection error for fragile mark first, and then from among

the keys, which result in the minimum fragile mark detection error, select the one, which results in the minimum robust mark detection error.

7. Given image partitioning block size b and image permutation key k, embed W_f and W_r into C using Standard two-level watermarking algorithm with image pixels selected based on the SNDM scores calculated in each pixel neighborhood of size n from set N, obtaining $\tilde{C}_n, n = 3x3,5x5,7x7$

8. Calculate visible distortion $DRDN_n\left(C,\tilde{C}_n\right), n = 3x3,5x5,7x7$

9. Find n which minimizes visible distortion, $n = \min_{3x3,5x5,7x7}\left(DRDN_n\left(C,\tilde{C}_n\right)\right)$.

10. $\tilde{C}_n$ is the watermarked image obtained by embedding self-verifying 12-bit message V into the C, followed by robust and fragile marks W_r, W_f.

The adaptive two-level watermarking scheme will detect and extract the embedded watermarks using the following algorithm:

1. Let $\tilde{C}$ be a watermarked image;
   Let B be a set of valid image partitioning block sizes B={b_i | $i=1,...m$};
   Let K be a set of valid image permutation keys (i.e the set of valid pseudo-random generator seeds), K = {k_j | $j=1,...n$};

2. Use valid image partitioning sizes from B, and valid image permutation keys from K to extract the first 12 bits from the watermarked image $\tilde{C}$.

3. The pair (b,k) which extracts the self-verifiable 12-bit sequence image is the image partitioning block size and the image permutation key which should be used for watermark detection.

4. Use the image partitioning block size b and the image permutation key k to extract robust and fragile watermarks W_r *and* W_f .

The results of the adaptive two-level watermarking scheme applied to the CCITT6 binary document image are illustrated in Figure 14.15-Figure 14.17.

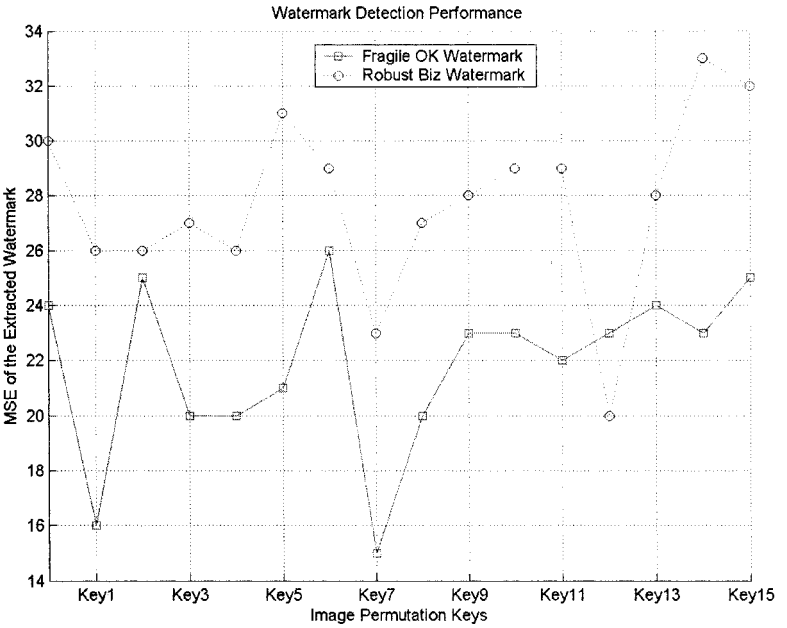


Figure 14.15 Mean Squared Errors of the extracted Biz and OK watermarks change as the image permutation key used for watermark embedding changes.

Figure 14.15 shows how the watermark detection error measured by MSE changes as different image permutation keys are used as part of the watermark embedding and detection process. Different image permutation keys affect watermark detection error because they change a distribution of image modification candidate pixels across image partitioning blocks. The use of permutation key 207, for example, results in the smallest watermark detection errors in the presented set of data. The results of embedding Biz and OK marks into the CCITT6 binary document image using the permutation key 207 are illustrated in Figure 14.16. The observed quality of watermarked image is high. This is confirmed by calculating DRDM=0.098. The calculated watermark detection error is low for both marks, MSE(Biz)=23, and MSE(OK)=15.

The use of permutation key 215 results in the largest watermark detection errors in the presented set of data, and the results of embedding Biz and OK marks into the CCITT6 binary document image using it are illustrated in Figure 14.17.

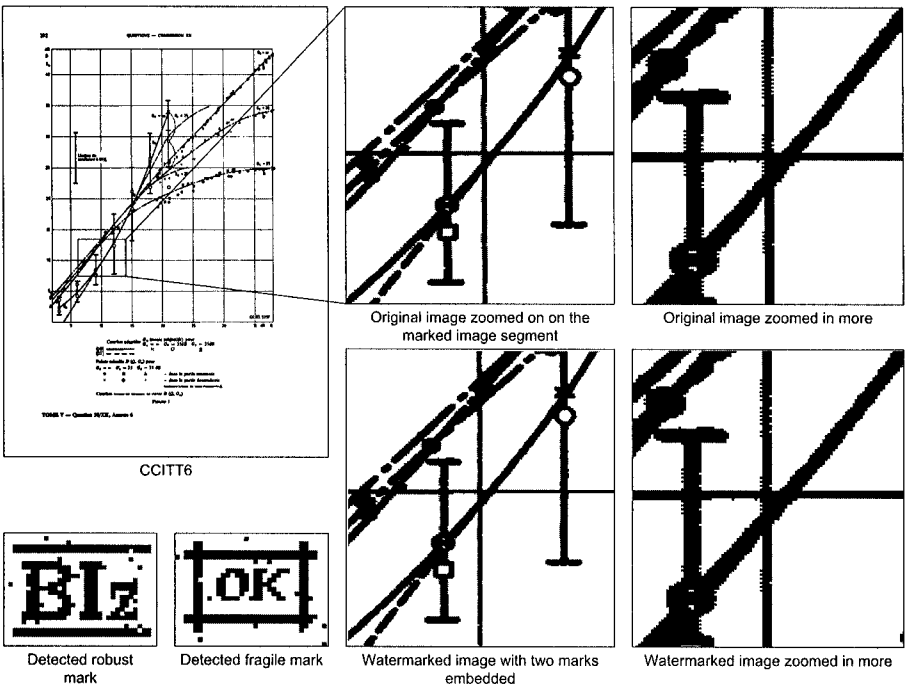


Figure 14.16 Result of embedding Biz and OK marks into the CCITT6 binary document image using image permutation key 207.

The quality of watermarked image is high, DRDM=0.0083, but the watermark detection error is much higher than the watermark detection error when the embedding was based on the image permutation key 207. With the permutation key 215, watermark detection error for the robust mark is MSE(Biz)=32, and for the fragile mark isMSE(OK)=25.

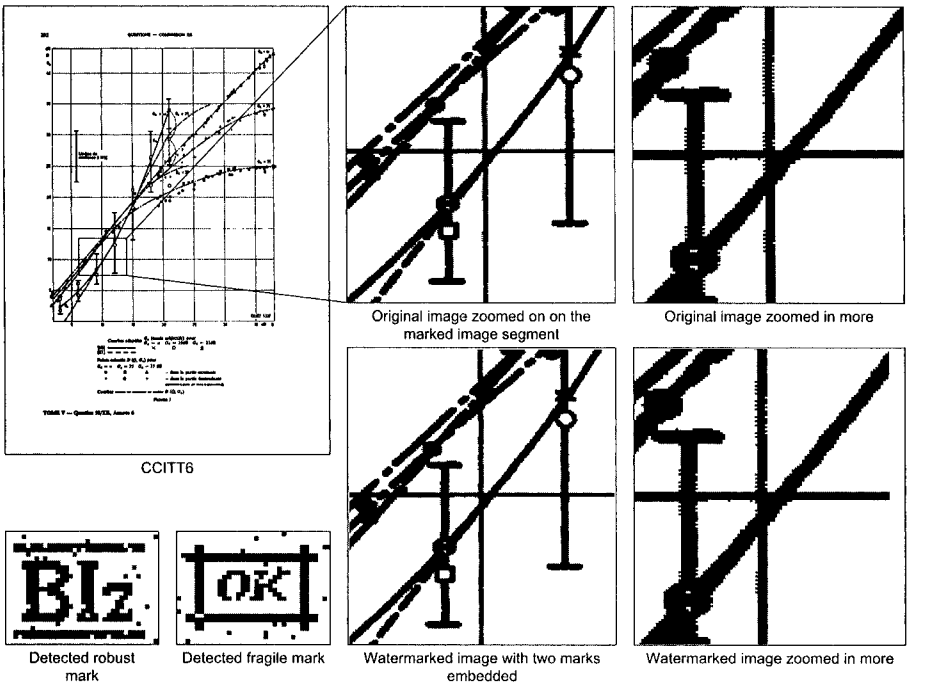


Figure 14.17. Result of embedding Biz and OK marks into the CCITT6 binary document image using image permutation key 215.

Chapter 15

FUTURE OF WATERMARKING

Digital watermarking field has grown to reach its teenage years. It was born as a result of desire to find solution for the copyright protection of digital multimedia content and offer technology which could be used to control copying and distribution of copyrighted material. Copyright protection was seen as a "killer application", and potential use of watermarking technology as its enabler triggered a lot of interest in both industry and research community. During its initial years, digital watermarking has become one of the most active research fields in the area of image and signal processing judging by the number of published papers on watermarking presented at all major conferences throughout the world.

Unfortunately the requirements of copyright protection and more generally security-oriented applications are very difficult to satisfy. The initial expectations of watermarking turned out to be too high, and the level of security required by copyright protection applications was out of reach of watermarking technology. When the watermarking technology failed to deliver an appropriate solution for the copyright protection problem, it appears that not enough trust was left to explore other possible useful applications of the developed technology, causing the interest in digital watermarking to start fading, and funding for watermarking research, especially in Europe, to start decreasing.

However, watermarking technology can still be successfully applied in many applications other than security-related ones, and watermarking can still provide economic solutions to the real-world problems.

Current business interest is focused on applying watermarking technology to provide new and enhanced services which can be delivered over the existing infrastructure. For example, it is possible to deploy new services by adding digital watermark data into the

existing legacy channels, such as NTSC TV signal or AM/FM radio signal, while remaining compatible with the pre-existing services and equipment. Or, audio data can be embedded into an existing image database. Database images are still stored in standard formats and they are still accessible using standard browsers. The embedded audio data can be accessed as a special service using special application or device.

Error concealment represents another useful application of watermarking technology. The watermarked bit stream remains standard compliant for compatibility with the existing equipment, and players which can exploit the embedded information can deliver higher quality media because they will have lower bit error rate.

Those approaches make it possible to offer enhanced services without affecting the functionality of existing systems. New solutions based on watermarking can coexist with the present solutions and they offer the enhanced performance of products available now. Watermarking technology can thus be used as a good tool for transitioning from one standard to another, especially when not all infrastructure of a particular standard can be easily upgraded. Those are solutions which depend on the large installed base of legacy services, where the cost of replacing the infrastructure used by an existing service may be too high.

Watermarking is also deployed in environments where there is no active or malicious adversary to cope with, such as linking media to the Web to provide various system enhancements.

Finally, even though many proposed watermarking solutions were criticized as not being robust enough because it was relatively easy to circumvent the security system, there are still many business applications where any level of security is better than no security at all. Just because a security protection system can be rendered inoperable does not mean that consumers will do it. The Macrovision's analog protection system can be used to demonstrate that this is true, because even though the analog protection system provided only limited copy protection of VHS tapes, studios continued to use the Macrovision's services. Similarly, watermarking can have its value and purpose when applied as a tool to increase threshold for illegal actions, and be applied in copy protection and fingerprinting applications.

In conclusion, a lot of progress has been made in the development of the core watermarking technology. The technology is being slowly adopted by the industry, and new commercial applications based on watermarking technology are being developed. Broadcast monitoring and content authentication applications based on digital watermarking are leading the way, and variety of new applications are emerging which enhance existing systems by either adding additional service or linking the content to the Web.

REFERENCES

[1] Ahumada, A. J., and Peterson, H. A. "Luminance-model-based DCT quantization for color image compression," *in Proc. SPIE*, vol. 1660, (1992): 365-374.

[2] Amamo, T., and Misaki, D. "Feature calibration method for watermarking of document images," *in Proc. of 5th Intl. Conf. on Document Analysis and Recognition*, (1999): 91-94.

[3] Arnold, M., Schmucker, M, and Wolthusen, S.D. *"Techniques and Applications of Digital Watermarking and Content Protection"*, Artech House, (2003).

[4] Bell, A.E. "The dynamic digital disk," IEEE *Spectrum*, vol. 36(10), (October 1999): 28 -35.

[5] Bender, W., Gruhl, D., Morimoto, N., and Lu A. "Techniques for Data Hiding," *IBM Systems Journal*, vol. 35, no. 3&4, (1996): 313-336.

[6] Berrou, C. and Glavieux, A. "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44(10), (October 1996): 1261 -1271.

[7] Bloom, J.A., Cox, I.J., Kalker, T., Linnartz, J.-P.M.G.. Miller, M.L., and Traw, C.B.S.; "Copy Protection for DVD Video," in *Proc. of the IEEE*, vol. 87(7), (July 1999): 1267 -1276.

[8] Boneh, D., and Shaw, J. "Collusion-secure fingerprinting for digital data," *IEEE Transactions on Information Theory*, vol. 44(5), (September 1998): 1897-1905

[9] Burgett, S., Koch, E., and Zhao, J. "Copyright labeling of digitized image data", IEEE *Communications Magazine*, vol. 36(3), (March 1998): 94-100.

[10] Chen, M., Wong, E.K., Memon, N. and Adams, S. "Recent developments in document image watermarking and data

hiding," *in Proc. SPIE Conf. 4518: Multimedia Systems and Applications IV*, (August 2001): 166-176.

[11] Chen, B., and Sundberg, C.-E.W. "Digital audio broadcasting in the FM band by means of contiguous band insertion and pre-canceling techniques," *IEEE Transactions on Communications*, vol. 48(10), (October 2000): 1634 -1637.

[12] Chen, B., and Wornell, G.W. "An information-theoretic approach to the design of robust digital watermarking systems," in Proc. of *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, (March 1999): 2061 - 2064.

[13] Chen, B., and Wornell, G.W. "Quantization index modulation: a class of provably good methods for digital watermarking and information embedding," *IEEE Transactions on Information Theory*, vol. 47(4), (May 2001): 1423 -1443.

[14] Chou, J., Pradhan, S.S., El Ghaoui, L., and Ramchandran, K.; "Watermarking based on duality with distributed source coding and robust optimization principles," *in Proc. of Intl. Conference on Image Processing*, vol. 1, (September 2000): 585-588

[15] Costa, M. "Writing on dirty paper," *IEEE Transactions on Information Theory*, vol. 29(3) , (May 1983): 439 -441.

[16] Cox, I.J.; Miller, M.L.; Bloom, J.A., *"Digital Watermarking,"* Academic Press, Morgan Kaufmann, (2001).

[17] Cox, I.J., Kilian, J., Leighton, F.T., and Shamoon, T. "Secure spread spectrum watermarking for multimedia," *IEEE Transactions on Image Processing*, vol. 6(12), (December 1997): 1673 -1687.

[18] Cox, I.J., and Linnartz, J.-P.M.G. "Some general methods for tampering with watermarks," *IEEE Journal on Selected Areas in Communications*, vol. 16(4), (May 1998): 587 -593.

[19] Cox, I.J., and Miller, M.L. "Electronic watermarking: the first 50 years," *IEEE Fourth Workshop on Multimedia Signal Processing*, (October 2001): 225-230.

[20] Cox, I.J., Miller, M.L., and McKellips, A.L. "Watermarking as communications with side information," in *Proc. of the IEEE*, vol. 87(7), (July 1999): 1127-1141.

[21] Craver, S., Memon, N., Yeo, B.-L., and Yeung, M.M.; "Resolving rightful ownerships with invisible watermarking techniques: limitations, attacks, and implications," *IEEE Journal on Selected Areas in Communications*, vol. 16(4), (May 1998): 573 - 586.

[22] Craver, S., Memon, N., Boon-Lock Yeo, and Yeung, M.M. "On the invertibility of invisible watermarking techniques," *in Proc. of Intl. Conference on Image Processing*, vol. 1, (October 1997): 540 -543.

[23] Craver, S.A., Wu, M., and Liu, B. "What can we reasonably expect from watermarks?," *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*, (October 2001): 223 -226.

[24] Decker, S. "Engineering considerations in commercial watermarking," IEEE *Communications Magazine*, vol. 39(8), (August 2001): 128 -133.

[25] Depovere, G., Kalker, T., and Linnartz, J.-P.; "Improved watermark detection reliability using filtering before correlation," in Proc. of *Intl. Conference on Image Processing*, vol. 1, (October 1998): 430-434.

[26] Deseilligny, M. Pierrot, and Le Men, H. "An algorithm for digital watermarking of binary images, application to map and text images," IGN/DT/MATIS, France.

[27] Dugelay, J.-L., and Roche, S. "Fractal transform based large digital watermark embedding and robust full blind extraction," *in Proc. of IEEE Intl. Conference on Multimedia Computing and Systems*, vol. 2, (June 1999): 1003-1004.

[28] Eggers, J.J., Bauml, R., Tzschoppe, R., and Girod, B. "Scalar Costa Scheme for Information Hiding", *IEEE Transactions on Signal Processing*, vol. 51, no. 4, (April 2003).

[29] Eggers, J.J., Su, J.K., and Girod, B.; "Robustness of a blind image watermarking scheme," *in Proc. of 2000 Intl. Conference on Image Processing*, vol. 3, (September 2000): 17-20.

[30] Fisher, R. A., and Yates, F. *Statistical Tables*, London, 1938. Example 12.

[31] Franti, P. and Ageenko, E.I. "On the use of context tree for binary image compression," *in Proc. ICIP*, (October 1999): 752-756.

[32] Fridrich, J. "Robust bit extraction from images," *IEEE International Conference on Multimedia Computing and Systems*, vol. 2, (1999.): 536 -540.

[33] Friedman, G.L., "The Trustworthy Digital Camera: Restoring Credibility to the Photographic Image," *IEEE Transaction on Consumer Electronics*, vol. 39, no. 4, (November 1993): 905-910.

[34] Fu, M.S. and Au, O.C. "Data Hiding Watermarking for Halftone Images," *IEEE Transactions on Image Processing*, vol.11, no. 4, (April 2002): 477-484.

[35] Furht, B., and Muharemagic, E. "Multimedia Security: Watermarking Techniques," *IEC Comprehensive Report on Information Security*, International Engineering Consortium, Chicago, IL, (2003).

[36] Funk, W. and Schmucker, M. "High Capacity Information Hiding in Music Scores," *in Proc. of First Intl. Conf. on WEB Delivering of Music (Wedelmusic 2001), Florence, Italy*, (November 2001): 12-19.

[37] Hartung, F., and Girod, B. "Digital Watermarking of MPEG-2 Coded Video in the Bitstream Domain," *in Proc. of the IEEE Intl. Conference on Acoustics, Speech and Signal Processing*, vol. 4, (April 1997): 2621-2624.

[38] Hartung, F., and Kutter, M. "Multimedia watermarking techniques," *in Proc. of the IEEE*, vol. 87(7), (1999.): 1079 - 1107.

[39] Hel-Or, H. Z. "Watermarking and Copyright Labeling of Printed Images," *Journal of Electronic Imaging*, vol. 10, no. 3, (2001): 794-803.

[40] Katzenseisser, S., and Petitcolas, F.A.P. *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House, Boston–London, (2000).

[41] Kim, H.Y. and Afif, A. "Secure Authentication Watermarking for Binary Images," *in Prec. of the XVI Brazilian Symposium on Computer Graphics and Image Processing*, (SIBGRAPI 2003).

[42] Kirovski, D.; Malvar, H.; "Robust spread-spectrum audio watermarking," Proceedings of *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Volume: 3, May 2001, Pages: 1345 -1348

[43] Knuth, D. E. *The Art of Computer Programming*, vol. 2, 3rd ed., Section 3.4.2, Algorithm P, page 145. Addison-Wesley, (1997).

[44] Koch, E. and Zhao, J. "Embedding robust labels into images for copyright protection", *in Proc. Intl. Congress on Intellectual Property Rights for Specialized Information, Knowledge & New Technologies, Vienna*, (Aug. 1995).

[45] Kolchin, V.F., Sevastyanov, B.A. and Chistyakov, V.P. *Random Allocation*, V.H. Winston & Sons, (1978).

[46] Kutter, M, and Petitcolas, F.A.P. "A Fair Benchmark for Image Watermarking Systems," *in Proc. On SPIE in Security and Watermarking of Multimedia Contents*, (January 1999.):226-239.

[47] Langelaar, G.C., Setyawan, I., and Lagendijk, R.L.; "Watermarking digital image and video data. A state-of-the-art overview," *IEEE Signal Processing Magazine*, vol. 17(5), (September 2000): 20-46.

[48] Liu Y.W and Smith, J.O. "Multiple watermarking: Is power sharing better than time sharing?," *Center for Computer Research in Music and Acoustics*, Stanford, CA

[49] Lu C.S. and Liao, H.Y.M. "Multipurpose watermarking for image authentication and protection," *IEEE Transactions on Image Processing*, vol. 10, no. 10, (2001): 1579-1592

[50] Lu, H. Wang, J. Kot, A.C. and Shi, Y.Q "An objective distortion measure for binary document images based on human visual perception," *in Proc. Int. Conf. on Pattern Recognition*, vol. 4, (August 2002): 239-242.

[51] Matsui, K. and Tanaka, K. "Video-steganography: how to secretly embed a signature in a picture," *in Proc. IMA Intellectual Property Project*, vol. 1, no. 1, (1994): 245-249.

[52] Maxemchuk, N.F. and Low, S.H. "Marking text documents," *in Proc. IEEE Intl. Conf. on Image Processing*, (October 1997).

[53] Mei, Q. , Wong, E.K., and Memon, N. "Data hiding in binary text documents," *in Proc. SPIE Conf.: Security and*

Watermarking of Multimedia Contents III, (January 2001): 166-176.

[54] Mintzer F. and Braudaway, G. "If one watermark is good, are more better?," *in Proc. Int. Conf. on Acoustics, Speech, Signal Processing (ICASSP)*, vol. 4, (March 1999): 2067-2069.

[55] Moulin, P., and O'Sullivan, J.A. "Information-theoretic analysis of information hiding," *IEEE Transactions on Information Theory*, vol. 49(3), (March 2003): 563-593.

[56] Muharemagic, E., and Furht, B. "Survey of Watermarking Techniques and Applications", in Multimedia Security Handbook, CRC Press, (2004).

[57] Nikolaidis, A., Tsekeridou, S., Tefas, A., and Solachidis, V. "A survey on watermarking application scenarios and related attacks," *in Proc. of 2001 Intl. Conference on Image Processing*, vol. 3, (October 2001): 991-994.

[58] O'Ruanaidh, J.J.K., and Pun, T. "Rotation, scale and translation invariant digital image watermarking," *in Proc. of Intl. Conference on Image Processing*, vol. 1, (October 1997): 536 -539.

[59] Petitcolas, F.A.P. "Watermarking schemes evaluation," *IEEE Signal Processing Magazine*, vol. 17(5), (September 2000): 58 - 64.

[60] Pitas, I. "A method for signature casting on digital images," *in Proc. of Intl. Conference on Image Processing*, vol. 3, (1996.): 215 -218.

[61] Piva, A., Barni, M., Bartolini, F., and Cappellini, V.; "DCT-based watermark recovering without resorting to the uncorrupted original image," *in Proc. of Intl. Conference on Image Processing*, vol. 1, (October 1997): 520 -523.

[62] Podilchuk, C.I., and Delp, E.J. "Digital watermarking: algorithms and applications," IEEE *Signal Processing Magazine*, vol. 18(4), (July 2001): 33-46.

[63] Podilchuk, C.I., and Wenjun Zeng; "Image-adaptive watermarking using visual models," *IEEE Journal on Selected Areas in Communications*, vol. 16(4), (May 1998): 525-539.

[64] Pradhan, S.S., Chou, J., and Ramchandran, K. "Duality between source coding and channel coding and its extension to

the side information case," *IEEE Transactions on Information Theory*, vol. 49(5) , (May 2003): 1181-1203.

[65] Ramkumar, M., Akansu, A.N., and Alatan, A.A. "A robust data hiding scheme for images using DFT," *in Proc. of 1999 Intl. Conference on Image Processing*, vol. 2, (October 1999): 211-215.

[66] Ruanaidh, J.J.K.O., Dowling, W.J., and Boland, F.M.; "Phase watermarking of digital images" *in Proc. of Intl. Conference on Image Processing*, vol. 3, (September 1996): 239 -242.

[67] Tewfik, A.H.; "Digital watermarking," *IEEE Signal Processing Magazine*, vol. 17(5), (September 2000): 17-18.

[68] Tseng, Y.C., and Pan, H.K. "Data Hiding in 2-Color Images," *IEEE Transactions on Computers*, vol. 51, no. 7, (July 2002).

[69] Wang, Z., and Bovik, A.C. "A Universal Image Quality Index," *IEEE Signal Processing Letters*, vol. 9(3), (2002): 81-84.

[70] Watson,A.B. "DCT Quantization Matrices Optimized for Individual Images," *in SPIE Proc. on Human Vision, Visual Processing, and Digital Display IV*, (1993).

[71] Wenjun Zeng, and Liu, B. "On resolving rightful ownerships of digital images by invisible watermarks," *in Proc. of Intl. Conference on Image Processing*, vol. 1, (1997): 552 -555.

[72] Wenwu Zhu, Zixiang Xiong, and Ya-Qin Zhang. "Multiresolution watermarking for images and video: a unified approach," *in Proc. of 1998 Intl. Conference on Image Processing*, vol. 1, (October 1998): 465-468.

[73] Wolfgang, R.B., and Delp, E.J. "A watermark for digital images," *in Proc. of Intl. Conference on Image Processing*, vol. 3, (1996): 219 -222.

[74] Wolfgang, R.B., Podilchuk, C.I., and Delp, E.J. "Perceptual watermarks for digital images and video," *in Proc. of the IEEE*, vol. 87(7), (July 1999): 1108-1126.

[75] Wong, P.H.W. and Au, O.C. "A capacity estimation technique for JPEG-to-JPEG image watermarking", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13(8) , (August 2003): 746 -752.

[76] Wong, P.W. "Image Quantization, Half-toning, and Printing," *in Handbook of Image & Video Processing*, Editor Al Bovik, Academic Press, (2000).

[77] Wu M., and Liu, B. "Watermarking for image authentication," *in Proc. IEEE Int. Conf. on Image Processing*, (ICIP 1998).

[78] Wu M., and Liu, B. "Data hiding in image and video: Part I – Fundamental Issues and Solutions," *IEEE Transactions on Image Processing*, vol. 12, no. 6, (June 2003).

[79] Wu, M., Yu H., and Liu, B. "Data hiding in image and video: Part II – Designs and Applications," *IEEE Transactions on Image Processing*, vol. 12, no 6, (June 2003).

[80] Wu, M., Tang, E., and Liu, B. "Data hiding in digital binary images," *in Proc. IEEE Int'l Conf. on Multimedia and Expo*, (August 2000).

Index

D

E

F