Media Security

Andreas Uhl

Department of Computer Sciences
University of Salzburg

October 14th, 2014

Overview

1. Formalia
2. Introduction & Motivation
3. Encryption of Visual Data
   - Basics
   - Pay TV Systems
   - DVD Security
   - BLU RAY Security
   - Partial Encryption Schemes
4. Information Hiding & Watermarking
   - Robust Watermarking
   - Steganography
   - Integrity Verification for Visual Data
5. Literature

Outline

1. Formalia
2. Introduction & Motivation
3. Encryption of Visual Data
   - Basics
   - Pay TV Systems
   - DVD Security
   - BLU RAY Security
   - Partial Encryption Schemes
4. Information Hiding & Watermarking
   - Robust Watermarking
   - Steganography
   - Integrity Verification for Visual Data
5. Literature

Lecturer

Email-Address: `uhl@cosy.sbg.ac.at`.

Basis-URL: `http://www.cosy.sbg.ac.at/˜uhl`.

Office: FB Computerwissenschaften (Department of Computer Sciences), Room 1.15, Jakob-Haringer Str. 2, Salzburg-Itzling.

Telefon (Office): (0662) 8044-6303.

Telefon (Secretary): (0662) 8044-6328 or -6343.

When & Where

Course-URL: http://www.cosy.sbg.ac.at/~uhl/student.html.

When: Di 8:30 - 10:00

Interval: weekly

Where: Lecture Room T02

This lecture & Exam

Welcome to the lecture on “Media Security”. This lecture is of overview-type but still covers lots of research-related material since the subject-area is rather a recent one.
Multimedia Security has become a rather larger field, the focus of this lecture emphasises the topics covered locally, especially targeting visual media types (i.e. images, video, 3D data).
We offer 3 variants for the exam:

1. Classical exam (orally)
2. When signing attendence lists: 2 papers according to students interest and some basic knowledge about lectures’ content
3. When signing attendence lists: Extending the projects of the lab (Proseminar) and some basic knowledge about lectures’ content

Outline

1. Formalia
2. **Introduction & Motivation**
3. Encryption of Visual Data
   - Basics
   - Pay TV Systems
   - DVD Security
   - BLU RAY Security
   - Partial Encryption Schemes
4. Information Hiding & Watermarking
   - Robust Watermarking
   - Steganography
   - Integrity Verification for Visual Data
5. Literature

What’s so special about MultiMedia Data ?

Classical Cryptography provides well investigated techniques for all classical security applications. On the other hand, we have seen examples of weak multimedia security systems in the past – DVD CSS, SDMI – but also successful deployments like DRM systems – I-Tunes – or fingerprinting schemes (e.g. traitor tracing in the Oscar academy). What’s different then compared to classical cryptography usage ?

- Data volume (e.g. HDTV) and possible real time constraints.
- Perceptual aspects in MultiMedia (e.g. a .png, a .gif, and a .jpg file may be displayed or perceived entirely identical whereas the corresponding binary representation is highly different).
- Entirely different functionalities which cannot be met by classical cryptography (e.g. transparent encryption in a try and buy scenario, information hiding/data embedding applications, robust hashes and robust digital signatures).

Which fields/applications are covered by “Multimedia Security” ?

Cryptography marries Multimedia Signal Processing !!

- Media Encryption
- Information Hiding
  - Steganography
  - Robust Watermarking (Copyright, Annotation, Copy control, Fingerprinting, ...)
  - (Semi)Fragile Watermarking
- Data Integrity
  - (Semi)Fragile Watermarking
  - Perceptual Hashing
  - Media Forensics
- Biometrics

Top Journals

- IEEE Transactions on Information Forensics and Security (TIFS)
- EURASIP Journal on Information Security (JIS)
- Springer LNCS Transactions on Data Hiding and Multimedia Security

.... or in more general purpose Journals in the areas of Multimedia, Signal Processing and Security (“CRYPTOGRAPHY MARRIES MULTIMEDIA SIGNAL PROCESSING”).

International Conferences

- ACM Workshop on Information Hiding and Multimedia Security (2013 in Montpellier, 2014 in Salzburg, 2015 in Portland)
- IEEE International Workshop on Information Forensics and Security WIFS (2014 in Atlanta)
- SPIE's Media Watermarking, Security and Forensics (in the framework of Electronic Imaging in San Jose/San Francisco)
- Int. Workshop on Digital Watermarking and Forensics (2014 in Taipeh)
- Communications and Multimedia Security CMS (2014 in Alveiro)

Many more smaller meetings ..... and special sessions and special tracks at larger meetings in the areas of Multimedia, Signal Processing, Information Security.

Local Projects @ Wavelab

- Privacy-protected Video Surveillance on Scalable Bitstreams (FFG, with Commend International, 200K EUR)
- Biometric Sensor Forensics (FWF, 300K EUR)
- Structure-Preserving State-of-The-Art Video Watermarking (FFG, with SONY DADC, 100K EUR)
- Adaptive Security Techniques for Visual Data in Wavelet-based Representation (FWF, 250K EUR)
- Wavelet Compression for Video Surveillance Applications (Dallmeier Elektronik, 120K EUR)
- Adaptive Streaming of Secure Scalable Wavelet-based Video (FWF, 230K EUR)
- Sample Data Compression and Encryption in Biometric Systems (FWF, 210K EUR)
- ECRYPT - IST European Network of Excellence in Cryptology Watermarking Virtual Lab (EU, 50K EUR)

Literature

- Monographs
  - B. Furht and D. Kirovski. Multimedia Security Handbook. CRC Press, 2005.
  - W. Zeng, H. Yu, and C.-Y. Lin. Multimedia Security Techniques for Digital Rights Management. Elsevier Science, 2006.
  - A. Uhl and A. Pommer. Image and Video Encryption: From Digital Rights Management to Secure Personal Communications. Springer Series on Advances in Information Security vol. 15, 2005.
  - S. Katzenbeisser and F. Peticolas. Information Hiding Techniques for Steganography and Digital Watermarking. Artec House, 1999.
  - I. Cox, M. Miller, and J. Bloom. Digital Watermarking and Steganography. Academic Press, 2008.
  - R. Liu, W. Trappe, J. Wang, M. Wu, and H. Zhao. Multimedia Fingerprinting Forensics for Traitor Tracing. EURASIP Series on SP&V, Vol. 4, 2005.
  - J. Fridrich. Steganography in Digital Media. Camebridge University Press 2009.
  - H.T. Sencar, N. Memon. Digital Image Forensics – There is More to a Picture than Meets the Eye. Springer 2013.

Local Publications

- Journals
  - Dominik Engel, Thomas Stütz, Andreas Uhl. A survey on JPEG2000 encryption. ACM/Springer Multimedia Systems 15:4, 243-270, 2009.
  - Dominik Engel, Thomas Stütz, Andreas Uhl. Format-compliant JPEG2000 Encryption in JPSEC: Security, Applicability and the Impact of Compression Parameters. EURASIP Journal on Information Security, Article ID 94565, pp. doi:10.1155/2007/94565, 20 pages, 2007.
  - Hermann Hellwagner, Robert Kuschnig, Thomas Stütz, Andreas Uhl. Efficient In-Network Adaptation of Encrypted H.264/SVC Content. Signal Processing: Image Communication 24:9, 740-758, 2009.
  - Peter Meerwald, Christian Koidl, Andreas Uhl. Attack on 'Watermarking Method Based on Significant Difference of Wavelet Coefficient Quantization'. IEEE Transactions on Multimedia 11:5, 1037-1041, 2009.

Local Publications

- Journals
  - Gerold Laimer, Andreas Uhl, Key Dependent JPEG2000-Based Robust Hashing for Secure Image Authentication, EURASIP Journal on Information Security, Article ID 895174, doi:10.1155/2008/895174, 19 pages, 2008.
  - Dominik Engel, Elias Pschernig, Andreas Uhl. An Analysis of Lightweight Encryption Schemes for Fingerprint Images. IEEE Transactions on Information Forensics and Security 3:2, pp. 173-182, 2008.
  - Thomas Sttz, Florent Autrusseau, Andreas Uhl. Non-blind structure-preserving substitution watermarking of H.264/CAVLC inter-frames. IEEE Transactions on Multimedia 16:5, pp. 1337-1349, 2014.
  - Stefan Jenisch, Andreas Uhl. A detailed evaluation of format-compliant encryption methods for JPEG XR-compressed images. EURASIP Journal on Information Security 2014:6, 2014.

Outline

1 Formalia

2 Introduction & Motivation

3 Encryption of Visual Data
- Basics
- Pay TV Systems
- DVD Security
- BLU RAY Security
- Partial Encryption Schemes

4 Information Hiding & Watermarking
- Robust Watermarking
- Steganography
- Integrity Verification for Visual Data

5 Literature

What makes Encryption of Visual Data Special ?

Advise of the classical cryptographer: “No matter which data needs to be protected, take a strong cipher and encrypt the data. There is no alternative to that.”

Is there anything wrong with that statement when applied to multimedia data ? There might exist reasons to handle these data types in specific manner ....

- extreme data volume (e.g. high definition video), real time constraints
- different requirements with respect to the required security level (e.g. VoD, video conferencing, TV-News Broadcast, medical imagery) and different “value” of the data (if costs for attacking the encryption are higher than the value of the data the attacker will purchase access rights)

Contd ......

What makes Encryption of Visual Data Special ?

- different “importance” of the attackers (e.g. copying DVDs for private use is no problem but for redistribution and reselling it is of course) and how to distinguish among them
- QoS when data are transferred over networks which might require rate adaptation, transcoding; streaming of data and transmission errors

Obviously there are lots of questions associated with encrypting multimedia data and it seems to be not that simple as it seems at first sight. Entirely different functionality (e.g. transparent encryption) is possible as well.

Example: Medical Applications

The organisation of todays health systems often suffers from the fact that different doctors do not have access to each others patient data. The enormous waste of resources for multiple examinations, analyses, and medical check-ups is an immediate consequence. In particular, multiple acquisition of almost identical medical image data and loss of former data of this type has to be avoided to save resources and to provide a time-contiguous medical report for each patient. A solution to these problems is to create a distributed database infrastructure where each doctor has electronic access to all existing medical data related to a patient, in particular to all medical image data acquired over the years. Classical Telemedicine adds interactivity to that.
There is urgent need to provide and protect the confidentiality of patient related medical image data when stored in databases and transmitted over networks of any kind.

Example: Video Conferencing

In todays communication systems often visual data is involved in order to augment the more traditional purely audio-based systems. Whereas video conferencing (VC) has been around to serve such purposes for quite a while and is conducted on personal computers over computer networks, video telephony is a technology that has been emerging quite recently in the area of mobile cell phone technology.
No matter which technology supports this kind of communication application, the range of possible content exchanged is very wide and may include personal communication among friends to chat about recent developments in their respective relationships as well as video conferences between companies to discuss their brand-new product placement strategies for the next three years. In any case, each scenario requires the content to be protected from potential eavesdroppers for obvious reasons.

Example: Surveillance

The necessary protection of public life from terroristic or criminal acts has caused a tremendous increase of surveillance systems which mostly record and store visual data. Among numerous applications, consider the surveillance of public spaces (like airports or railway stations) and casino-gambling halls. Whereas in the first case the aim is to identify suspicious criminal persons and/or acts, the second application aims at identifying gamblers who try to cheat or are no longer allowed to gamble in that specific casino.

In both cases, the information recorded may contain critical private informations of the persons recorded and need to be protected from unauthorised viewers in order to maintain basic citizens' rights. This has to be accomplished during two stages of the surveillance application: first, during transmission from the cameras to the recording site (e.g. over a firewire or even wireless link), and second when recording the data onto the storage media.

Visual Example: Privacy & Surveillance

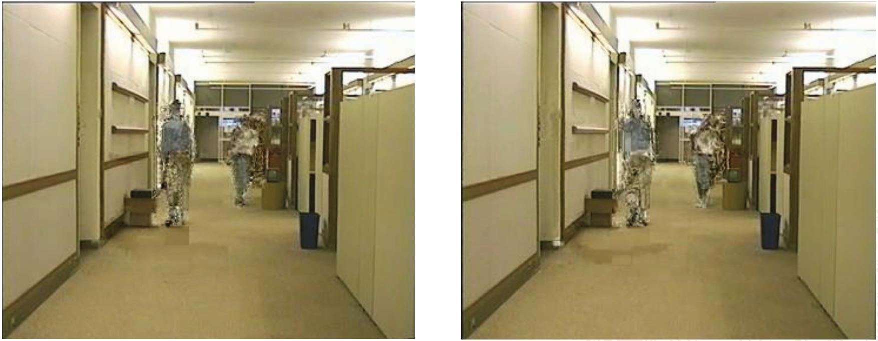

(a) permutation (b) sign bit encryption

Figure: Visual example for selective protection of privacy-relevant information.

Example: Video on Demand

VOD is an entertainment application where movies are transmitted from a VOD server to a client after this has been requested by the client, usually video cassette recorder (VCR) functionalities like fast forward or fast backward are assumed (or provided) additionally. The clients' terminals to view the transmitted material may be very heterogeneous in terms of hardware capabilities and network links ranging from a video cell phone to a HDTV station connected to a high speed fibre network.

To secure the revenue for the investments of the VOD company, the transmitted movies have to be secured during transmission in order to protect them from non-paying eavesdropping "clients" (encryption), and additionally, some means are required to disable a legitimate client to pass over the movies to a non-paying friend or, even worse, to record the movies, burn them onto DVD and sell these products in large quantities (watermarking and fingerprinting). Similar challenges have to be met with the DVD system ;-)

Consider a heterogeneous network with varying bandwidth, with transitions between wired and wireless. How to facilitate e.g. rate adaptation without decryption, transcoding, and re-encryption ?

Example: Pay-TV News

Free-TV is financed via commercials (everywhere) and/or via governmentally imposed, tax-like payments (like e.g. in Austria where everybody who owns a TV-set has to pay those fees no matter if he watches federal TV channels or not). Contrasting to that, Pay-TV is financed by the subscription payments of the clients. As a consequence, only clients having payed their subscription fees should be able to consume Pay-TV channels. This is usually accomplished by encryption of the broadcasted content and decryption in the clients' set-top box, involving some sort of smartcard technology.

Whereas the same considerations apply as in the case of VOD with respect to protecting the content during transmission, there is hardly any threat with respect to reselling news content to any other parties since news data loose their value very quickly.

Application-driven Media Encryption Security Levels

- Cryptographic Security: no information about the plaintext shall be deductible from the ciphertext. E.g., this includes indistguishability under a chosen plaintext attack (IND-CPA): Given two plaintexts and a corresponding ciphertext, an adversary cannot identify the correct plaintext with probability better than 0.5.
- Content security/Confidentiality: Information about the plaintext may leak, but the video content must not be descernible.
- Sufficient encryption: The content must not be consumable with pleasant viewing experience due to high distortion, but it is acceptable that content is descernible.
- Transparent/perceptual encryption: A preview image needs to be decodeable, but a higher quality version has to be protected.

Which applications would you assign to these security levels ? This depends on which aims you want to achieve and whom you want to protect e.g. VoD: content provider revenues vs. customer privacy !

Example: Request for Cryptographic Security in VoD

We have split well-known CIF sequences (Akiyo, Bus, City, Coastguard, Container, Crew, Flower, Football, Foreman, Harbour, Ice, Mobile, News, Silent, Soccer, Tempete, Waterfall) into non-overlapping subsequences of 8 frames. This results in 582 distinct sequences, of which some are very similar, e.g., the subsequences of the Akiyo sequence. The bitstreams were generated using the Joint Scalable Video Model (JSVM) 9.14 software. The scalable bitstream contains a QCIF substream (compliant to the H.264 baseline profile) and two CIF MGS layers to enable bitrate adaptation.

In order to assess the similarity between packet traces, the mean squared error of the packet lengths is considered.

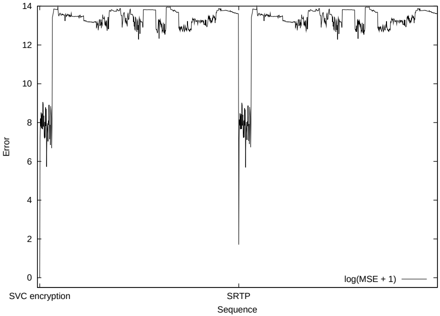
Example: Request for Cryptographic Security in VoD

If the number of packets differs between two packet traces, the MSE is calculated for the smaller number of packets. Thus, the difference between two packet traces pt_1 and pt_2 is defined in the following way:

$$\mathrm{d}(pt_1, pt_2) = \sum_{i=1}^{min(n_1,n_2)} (l_{1i} - l_{2i})^2$$

The overall number of packets for pt_1 is n_1 and the corresponding packet lengths are denoted by l_{1i} and similarly for pt_2. Two different encryption modes are considered (SVC-specific and the SRTP encryption).

Surprising Result



Despite very similar subsequences, each packet trace was unique for both the SVC-specific and the SRTP encryption. This fingerprint is not just unique, but also a weak measure for sequence similarity (shows MSE between the packet trace of the first subsequence from the Akiyo sequence with SVC-specific encryption and all other subsequences, even those encrypted with SRTP).

Implementation Levels of Media Encryption

- File Level: Encryption is applied regardless of the file type and structure.
- Transport Level: Encryption is applied regardless of the content – packets of stream segments of the transport layer are encrypted (e.g. using IPSec, TLS, SRTP).
- Metaformat Level: Encryption is applied within the scope of a metaformat, such as the ISO base media file format. Approaches which employ bitstream descriptions (e.g. MPEG-21 gBSD) and encryption fall into this category.
- Codec format Level: Encryption directly applied at the bitstream level, usually applied to preserve codec-specific features like scalability.

Kerkhoff's Principle: Is also valid for Media Encryption ! Security of a Cryptosystem must not stem from the secrecy of the technology but from secret cryptographic key material !!!

Pay TV Systems: Types

- Analog Systems: No cryptographic protection, parts of the information are transmitted in non standard way (e.g. interchanged synchronization information). Decoding should be possible only with decoders approved by the broadcaster –> highly insecure.
- Hybrid Systems: The signal is transmitted according to a anolog TV standard (like PAL, SECAM, NTSC). This signal is digitally encrypted in a frame buffer. The decoder usually implements the required algorithm in some sort of hardware (mostly smart cards). Examples: VideoCrypt, Syster/Nagravision, EuroCrypt
- Digital Systems: Digital signal (mostly MPEG-2,4 or H.264) is modulated and transmitted in analog way. Encryption is similar to hybrid systems but uses advanced cryptography. Example: DVB-S, DVB-T

Pay TV Systems: Payment

- Pay per Channel
- Prebooked pay-per-view: Specific movies or time-slots are pre-booked and pre-payed
- Impulse pay-per-view: Chipcard with a specific amount of paid impulses which are reduced the more you watch
- Near VoD: Movies are selected in some pre-defined time interval

Pay TV Systems: Security Concept I

The primary security means is encryption. The decryption module is provided at the receiver side. In older systems, the security system was integrated in the decoder (expensive replacement when compromised), current systems mostly use smart cards. There are two main subsystems:

1. Scrambling system: Encryption and decryption using the control word (CW) provided by the access control system
2. Access control system:
    - transmission of CWs for the scrambling system in real time. transmission needs to be confidential (entitlement control messages ECM). CWs are encrypted with the system key implemented in the smart card.
    - administration of access rights of the users (entitlement management message EMM). Protected by e.g. Fiat-Schamir identification scheme (VideoCrypt) and public-key schemes (e.g. EuroCrypt).

Pay TV Systems: Security Concept II

Corresponding to the two components of the security architecture, we may distinguish two different types of messages.

1. ECM (Entitelment control message - for global informations): Transmission of new codeword (CWs) for decrypting the program and messages concerning global program conditions (e.g. PG informations, fees are required to be paid for this program part, etc.). CWs are encrypted with the system key which resides on the smart card and is identical for all users.
2. EMM (Entitlement management message - for individual informations): For changing individual permissions concerning program receivement on the receiver side – e.g. if a custumer did not pay, he/she is no longer entitled to view the program. These information are privacy sensitive and need to be protected (e.g. by using the Fiat-Schamir identification scheme in VideoCrypt and a public-key algorithm in EuroCrypt).

Pay TV Systeme: Blocking Decoders

How can dedoders get blocked in case payment was not received ?

1. Negative addressing: Deletion of authorisations/entitlements with EMMs, i.e. the security module get deactivated partially or entirely (pirate cards do not have this functionality implemented)
2. Positive addressing: Wrong/invalid keys are being transmitted to such addresses – secure also against pirate cards, but the distribution of invalid keys causes a high number of EMMs which is expensive.

Attacks against PayTV Systems

- Pirate systems: Manipulated decoder or smart card technology is used, since the hardware is protected by patents this is problematic from a layers' viewpoint.
- Attacks against the Scrambling System: in the past mostly guided brute-force attacks against permutations (see below). Anybody is allowed to do this, there is hardly a way for broadcaster to stop this. The only way is to go for stronger ciphers (as done in current schemes).
- Recording and distributing key material: since CWs are identical for all smart cards (and are encrypted identically) they can be recorded. Similary, an encrypted signal may be recorded by any recording device and decrypted later using the recorded CWs ("offline internet card sharing"). A PC emulates the smart card.

Nagravision/Syster (e.g. Premiere Analog)

- For PAL, SECAM und NTSC
- Basic principle: line shuffling in blocks consisting of 32 lines.
- After 256 fields (half resolution images using interlaced video) the shuffling is changed by a newly transmitted CW.
- The line shuffle (permutation) is controled by a random number generator, which proceeds through a list of 256 numbers in the range between 0 and 31.
- The CW is the seed of the generators in this case, which determines where to start in the list and how to leap through the list. This results in $256 \times 128 = 32768$ possible permutations.

Attack against Nagravision/Syster

- Assumption: the list of 256 numbers is known, otherwise the attack is more complex.
- All 32768 possible permutations are computed.
- Basic principle: if two neighbouring lines in the unencrypted image which are separated in the encrypted version are neighbours again after applying a certain permutation, than the correct permutation key has been used with high probability.
- A couple of line pairs is needed to obtain sufficient security of correctness.
- Lines which are neighbours in the original need to be similar.
- Judgement criterium: a small sum of luminance differences (pixel wise). Tradeoff between speed and accuracy:
  - the more line candidates, the more accurate result
  - the more pixels are considered per line, the more accurate result

VideoCrypt I + II (z.B. SKY)

- For PAL, SECAM and NTSC, developed by Thomson Consumer Electronics
- Each line in a frame is cut at a secret position and the resulting pieces are interchanged (cut and rotate)
- There are 256 admissible cut points; these need to be 12 - 15 % of the line width away from the frame's edge (why?)
- Several times per second a 32 byte massage is broadcast.
- Every 2.5 seconds a MAC generates 60 bit, which are used as seed for a pseudo-random number generator, which outputs 8 bit (the cut points).
- In the decoder, the smardcard generates the MAC output.
- The 32 byte messages contain EMMs and a signature with which the smardcard can verify its authenticity.

Attack against VideoCrypt

- Due to the similarity of adjacent lines all 256 cut points can be tested. Again, reliability as well as attack complexity is increased when using several lines instead of just a sngle one.
- In 1993, the hash funktion got public – until then, it was one of the algorithms secrets (it is assumed that information leaked directly at the development site). Until a new generation of the system / smartcard could be generated (which took about 1 year), many pirate cards were in use.
- Security: Almost identical to Syster
- A nice example of violating Kerckhoffs principle with some associated cost (i.e. non-paid subscription fees).

Digital Video Broadcast (DVB)

- DVB is orginally based on MPEG-2, now moving to H.264, which enables broadcast of 30+ digital TV channel on a single satellite channel
- DVB receiver (set-top box) consists of:
  - Cable/antenna plug
  - Receiver and demodulation
  - Error correction
  - Access control and decryption (optional)
  - MPEG demultiplexer
  - Decoder for video, audio, text
  - D/A conversion (only f. PAL, NTSC, ...)

DVB Security

- By analogy to hybrid systems, we have a two-fold security concept: Actual encryaption of the MPEG data (“descrambling”) and the protected trasnmission of ECMs and EMMs (“decryption”).
- For descrambling, a common algorithm has been standardised, composed of a 64Bit blockcipher and a stream cipher (common scrambling algorithm). Technical details are only available for hardware manufacturers under non-disclosure agreement.
- For decryption, no common solution could be agreed on (fear of pirated smart cards)
- In order to enable channels broadcast by different broadcasters with a sngle set-top box, two options do exist: Common Interface and Simulcrypt.

DVB: Common Interface (CI) und Simulcrypt

- Common Interface (CI): CI is a standardised slot in the set-top box for one or more PCMCIA cards (CAMs - conditional access modules). On the CAM, eventual alternative scrambling algorithms and software and key material for decryption are implemented.
- Simulcrypt: Broadcaster can transmit data for the decryption system suited for several systems. This works, in case the common scrambling algorithm is used and in case the set-top box vendor (a broadcaster in many cases) and your target broadcaster do agree on the systems in use.

DVD: Security Concept I

The overall concept of DVD is the concept of trusted hardware, i.e. it is assumed that digital data does not leave the protected environment of licensed hardware. This type of hardware obeys the imnposed rules, other type of hardware does not, nor do eventual reverse engineered software, which causes problems.

- Regional Codes: DVD-Video is not interchangeable on an international level. US movie industry defined 6 regions worldwide in February 1997. DVDs may carry a code (this is not mandatory) which prevents them from being played on a player with different code. This strategy enables studios to control when and in which version movies can be purchased in DVD (of course, after these have been played in cinemas).
  Regional codes are not based on crypto, these are just control bytes which can be ignored if the hardware / software is capeable of. In many players, regional codes may be changed. This “security feature” does not have any more value nowadays.

DVD Security Concept II

- APS (Macrovision): Analog copy protection to prevent video cassette recorders from recording DVD content. Can be switched on and off for parts of the movie during DVD production. This scheme exploits slow reaction of TV sets on changes in the incoming signal while VCRs react promptly. APS modifies the signal in a way that VCRs record the introduced distortions, while a TV set is not affected. As VCRs have disappeared, this in no longer of relevance.
- Serial Copy Generation Management System (CGMS): Is meant to prevent digital copies by embedding CGMS control informations (here, watermarking (WM) technology has been developed but finally not adopted but control bytes used):
  - copy never
  - no more copies (is already a copy)
  - copy one generation
  - copy freely

In order for CGMS to work properly, hardware has to obey to the rules.

DVD Security Concept III

- Digital Transmission Content Protection (DTCP): This is meant for preventing protocol attacks during communication between two DVD hardware elements (e.g. a DVD player and a TV, or a DVD player and DVD recorder etc.). DVD licensed devices perform mutual authentication, exchange keysm and establish an encrypted connection to transfer video data. E.g., a TV set may receive and display all videos, while a DVD recorder can only record data meant for being recorded (see CGMS). DTCP uses strong and established crypto.

  An additional security feature are system updates in the form of System Renewal Messages, which mostly communicate which devices can no longer be considered secure. There messages are transmitted either via new discs of broadcast and are further communicated via DTCP among participating devices. DTCP has been originally devleoped for IEEE 1394 (firewire).

DVD Security Concept IV

- Content Scrambling System (CSS): Data on the DVD are encrypted by CSS (which is a secret, non-public cipher consisting of three LFSR) and can therefore only be displayed (i.e. and decrypted before that) on DVD-licensed devices.
  For display, decryption is done (digital transmission of plaintext video is not permitted) using information provided by acquiring the DVD license. Each DVD player has its own unlock-key (which is encrypted) and each DVD disc has 400 different 5-byte CSS keys, encrypted with the unlock-keys of all DVD partners. This way, a key in a licenced player unlocks the CSS key which in turn decrypts the didcs content.
  As before, a digital 1:1 copy cannot be prevented using this approach, in case the required professinal hardware is available. The average user is prevented from copying, but not the professional one (which is kind of strange of course). CSS only works in case device producers follow the rules as imposed.

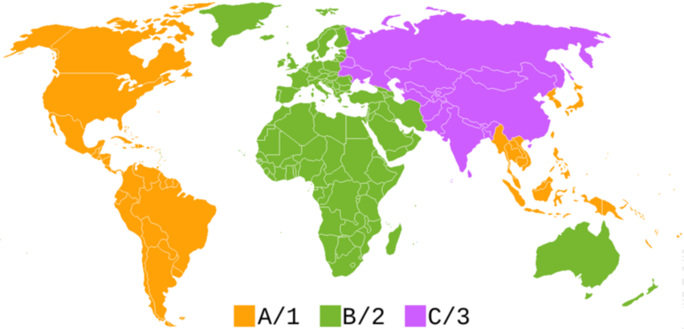
The DVD/CSS Hack

"We found that one of the companies had not encrypted their CSS decryption code, which made it very easy for us".
In the specific software player XingDVD (RealNetworks) the unlock-key was not properly encrypted (software only tried to conceal the key) and could therefore be read out. This enabled attackers to reverse-engineer the CSS code which was found some weeks later printed on T-shirts. It turned out that the employed cipher was a custom development by the DVD consortium exhibiting several weaknesses. Another nice example of violating Kerckhoffs prinicple !
In software players we always have the problem that key material shows up in some place in the registers of the executing processor, no matter how well protected it might be. Given enough compentence in this field, it can always be read out. Techniques to prevent this are called white-box crypography.
The CSS hack has delayed the introduction of writeable DVD !

BLU RAY Security Concept I

- Region Code
- High-bandwidth Digital Content Protection (HDCP)
- Advanced Access Content System (AACS)
- Self-Protecting Digital Content / BD+
- BD-ROM Mark

BLU RAY: Region Code



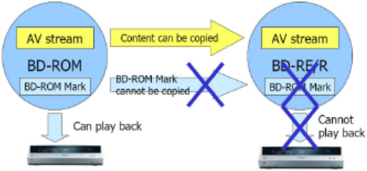
Similar to DVD region codes as these are only checked by the player software. Studios are free to decide to use this feature, about 70% of all titles are released without region code, but this strongly varies from studio to studio.

BLU RAY: HDCP

A BluRay disc can enforce to be displayed on lower resolution or even not to be displayed in case no valid HDCP-link has been established to the display device. This is similar to DVDs DTCP system. Before transmitting data, the authorisation of the receiving device is verified. In case of authorised transmission an encrypted channel is used.
Each HDCP device has a unique set of 40 keys with 56 bits. Additonally there is a KSV (key selection vector) consisting of 40 bits, one bit for each key. During device authentication KSV are exchanged and depending on the KSV of the other device, keys are added binary. This leads to an identical 56 bit key in both participating devices, which is used in a stream cipher to encrypt data during transmission.
In case of compromise, the KSV is burnt to the revocation list of new discs (digitally signed using DSA to prevent from revocing legitimate devices), which is checked during authentication.
The system has been broken in 2001, in 2010 a master key has been published neutralising the HDCP revocation system.

BLU RAY: BD-ROM Mark

BD-ROM mark is a small set of cryptographic data which is stored separately from other data (and contain the volume ID which is required for AACS decryption). When producing a copy of a disc, this data cannot be read-out. For data embedding, a licenced device is required.



The approach relies on inaccuracies in terms of revolution speed in the copying process. For pressed discs, the position of the data is maintained due to the precise production process, for copied discs the position is changed. The description of the expected position of the data on the disc is digitally signed and cannot be attacked therefore.

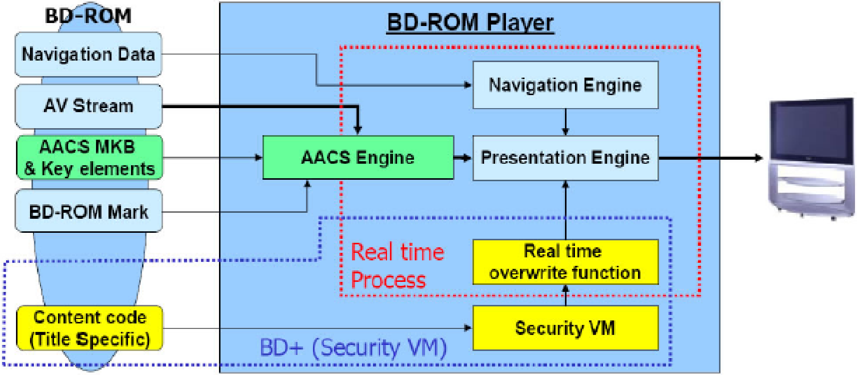
BLU RAY: Self Protecting Digital Content / BD+

BD+ is a virtual machine on BluRay players enabling to store and execute code on BluRay discs. This works as follows:

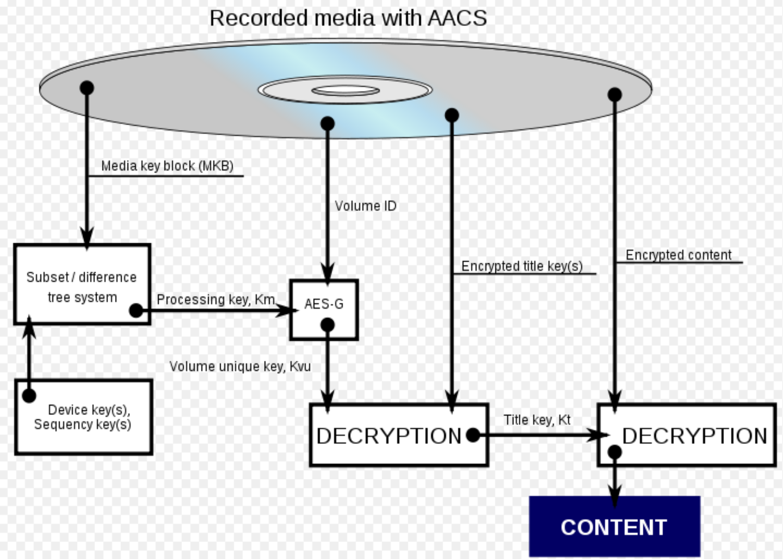
1. VM starts when inserting the disc
2. Eventual functions:
   - Checking of the player by comparing memory footprints to detect manipulations.
   - Checking the integrity of the players keys
   - Execute code to patch an insecure system (to be eventually provided by the manufacturer)
   - Transformation of audio and video data. Parts of the data can only be decoded by the BD+ system.
   - Correction / skipping of error-prone data parts
   - Insertion of WM (see later for the purpose of that).
3. After ejecting the disc, VM is temrinated and code is deleted form the players memory to re-establish its original state.

broken in 2008: Slysoft AnyDVD HD 6.4.0.0

BLU RAY: System View



BLU RAY: AACS Overview



BLU RAY: AACS I

AACS uses 128-bit AES encryption to protect video and audio on the BluRay. For decrypting, usually several “title keys” are used, which are encrypted by themselves. Title keys are decrypted using a combination of the “media key” (encoded in the “media key block”) and the volume ID of the disc (a pysical serial number embedded by the DB-ROM Mark. The result, the “volume unique key”, decrypts the title keys. In the MKB there are several media keys which are differently encrypted using a broadcast encryption scheme called “subset difference tree system” which specifically facilitated the revocation of single players. When changing the MKB, single or several players are disabled from playing the disc.

BLU RAY: AACS II

Sequence Key Blocks (SKB): Up to 32 short video segments can be encrypted using additional key material – in each of these 32 segments 8 variants exist out of which only a specific one can be decrpyated using the sequence keys. Different players thus decrypt different versions of the data, which delivers a fingerprint of the (insecure) player, which can then be revoked with the next MKB variant. The different versions can be distinguished by different WM embedded.

BLU RAY: Subset Differerence Tree System I

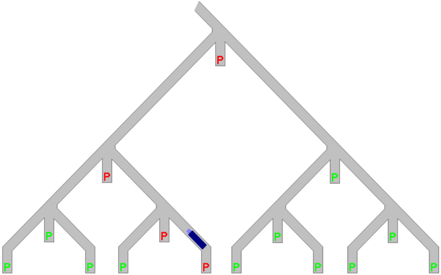
MKB is stored on the disc, the player holds a couple of “device keys” which are used to derive the “processing key” from the MKB, which is used in turn to decrypt the media key from a “C-value” (encrypted media key).

The MKB is a tree-like organised collection of encrypted processing keys and device keys. The latter are used to derive processing keys and since there is only a few device keys in the player, a single player is only able to derive a limited number of processing keys. This is also used to revoke single players in case of observed (SKB) insecurities.

In the following, we illustrate the process using lorries driving in a tree structure. Driving means the application of a one-way function (AES-G3) to the device key or a sub-device key which either results in the processing key or further sub-device keys.

BLU RAY: Subset Differerence Tree System II

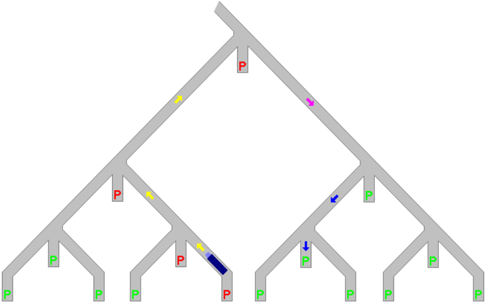
A lorry can take a curve with less degrees than 90 only and it is not able to reverse. Green parking lots can be reached but not red ones (from the lorry's starting position). The lorry is the player carrying some informations, parking lots are the processing keys in the MKB, which do not change, only the targeted parking lot may change (which is the required processing key ?).



MKB file contains the instructions which parking lot to target, where the information required to decipher the C (the encrypted media key) is to be found.

BLU RAY: Subset Differerence Tree System III

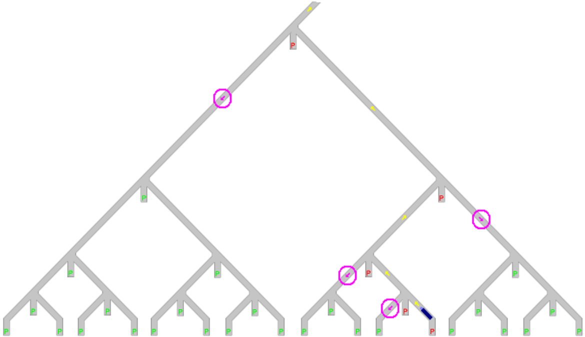
It is not possible to reach each parking lot. If a lorry has a starting position, from which the parking lot cannot be reached, the medai key cannot be produced. There are many lorries around – is a lorry (i.e. a player) to be revoked, a parking lot is chosen that cannot be reached by this lorry.



In the displayed example lorries drive northbound first, and turn to the south afterwards. In reality, (in AACS) lorries drive southbound only, lorries “jump” to the points from which they drive to the south. The points targeted by jumping are the device keys.

BLU RAY: Subset Differerence Tree System IV

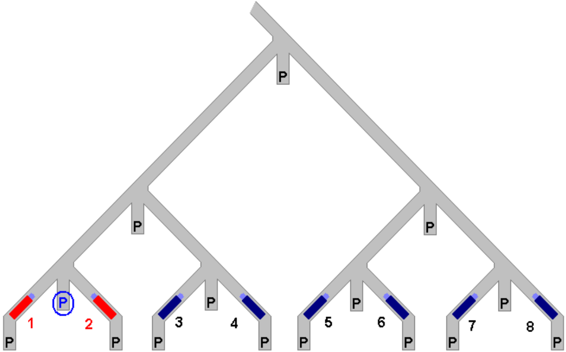
The image is a zoom - it is clearly visible, that there are several device keys enabling to drive to the south, however, only a single one is the correct entry point to reach the processing key.



In order to learn which device key has to be used, the MKB has to be consulted. Afterwards, the jump to the entry point in the tree can be conducted and the processing key is generated.

BLU RAY: Subset Differerence Tree System V

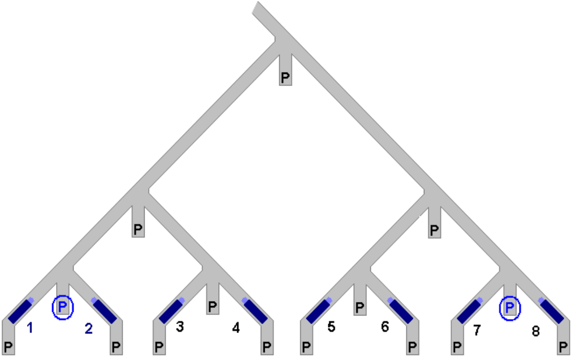
For revocation, processing keys are chosen which cannot be reached by specific players.



In the example, the blue encircled processing key is targeted, thus, lorry 1 & 2 are revoked. Can we also revoke non-neighbouring lorries or several ones ?

BLU RAY: Subset Differerence Tree System VI

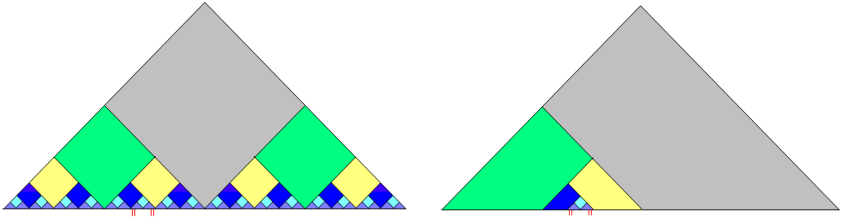
The idea is to revoke lorry 1,2,7,8 in the example by choosing two processing keys that cannot be reached.



This does not work as lorries can reach the more distant processing keys, thus, no player is revoked. This motivates, why it is not sufficient to have a single large tree.

BLU RAY: Subset Differerence Tree System VII

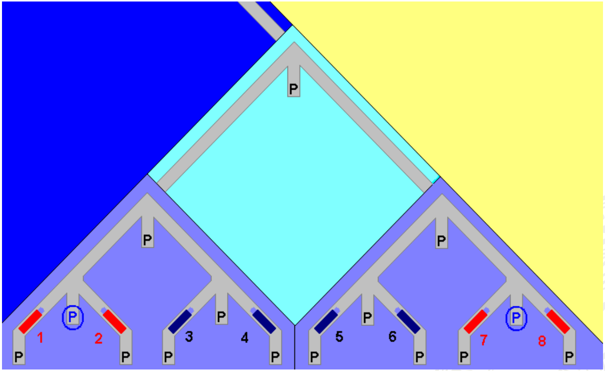
In fact, the solution is to have several tree layers (22 layers), which exhibit half the size of the previous layer when advancing in the tree upwards.



In the image is colour is a layer, the smaller layers are on top of the larger ones. The red dots are the positions of the lorries which should be revoked. Lorries can use an “elevator” for reaching their layer. The right image shows the part of the MKB relevant for the considered lorries.

BLU RAY: Subset Differerence Tree System VIII

In the image it is visible that there is no direct connection between the two sub-trees, now actually lorries 1,2,7,8 are revoked but 3,4,5,6 are not.



The different processing keys are actually different, which means that also the C-value corresponding to different parts of the tree has to be different to generate the identical media key.

Media Encryption: The Role of Compression

No matter if lossy or lossless compression is applied: compression needs ALWAYS be performed prior to encryption. The statistical properties of encrypted data do not allow any compression to be achieved and the data reduction of compression reduces the encryption effort.

1. On-line Applications: The processing chain leading to encryption starts with the plain image data (spatial domain), usually after aquisition of the data. Examples: video conferencing, digital cameras, surveillance applications.
2. Off-line Applications: The processing chain leading to encryption starts with a compressed bitstream. As soon as image data has been transmitted or stored once, they are usually represented in compressed form. Examples: visual data bases, photo CD-ROM, VoD. These applications are usually purely retrieval based.

Media Encryption: Lossless or Lossy Compression

- Lossy data formats achieve a much higher compression ratio - therefore, subsequent encryption effort is reduced. However, computational effort is usually much higher as compared to the lossless case.
- Why would you use lossless representations anyway:
  - The application does not allow any information loss (e.g. medical imaging, GIS data, maps, etc.)
  - The available hardware has not enough computational power to perform lossy compression.
  - The high bandwidth at the transmission channel or the high storage capacity available does not require lossy techniques to be applied.

Classification of Codec-format Level Media Encryption

1. Type of compression used: 8 × 8 pixel DCT, wavelet, wavelet packets, quadtree, ...
2. In which stage of the processing chain encryption is performed:
   - As an integral part of the compression stage – compression integrated encryption:
     - coefficient data (coefficient values, signs, etc.) are encrypted, permuted, etc.
     - secret compression settings (e.g. secret Huffman tables, secret transform domains)
   - After the compression stage – bitstream oriented encryption:
     - encryption of header data or payload data
     - scalable/embedded bitstreams: without bistream parsing
3. Application(aim) oriented: cryptographic security, content security, sufficient encryption, transparent encryption

Compression Integrated Encryption

Advantages

- It is much simpler to identify encryption relevant information in the transform domain as compared to the bitstream (transform domain coefficients vs. codewords in hex).
- Bitstream compliance is maintained automatically to some degree (depending on the type of compression of course).

Disadvantages

- Is the data already given as a bitstream, the data needs to be decompressed and recompressed if encryption takes place during compression – not suited for Offline scenarios.
- Existing compression hardware may not be used.
- Encryption takes place before certain satges of the compression pipeline – this threatens compression performance. Data subjected to encryption need to be selected carefully.

Bitstream oriented Encryption

Advantages

- Suited for Online and Offline scenarios – highly flexible.
- Encryption is decoupled from the costly compression stage.
- Standards like MPEG-4 IPMP and JPSEC follow this strategy.

Disadvantages

- Maintaining bitstream compliance can be very tedious (generation of marker sequences leading to undefined decoder behavious).
- Bitstream parsing to identify important parts can be time consuming (depending on the data format).

Partial / Selective Media Encryption

The development of lightweight encryption schemes is at the core of media encryption. Besides applying fast but weaker ciphers with certain drawbacks like permutations (also denoted as “soft encryption”), an alternative approach is to apply cryptographically strong ciphers to certain parts of the media data only. This can have two – in most cases contradicting – aims:

1. reduction of the computational effort by restricting the encryption to the perceptually (e.g. base layer or most important coefficients) or semantically (e.g. unique header data) most important parts of the data (tradeoff security - complexity).
2. preserving the bitstream structure by restricting the encryption to payload data parts while leaving header data unencrypted (in order to maintain format compliance for transcoding and error resilience, resynchronisation)

In the following, we focus at reduction of computational effort.

Partial / Selective Media Encryption: Application Scenarios

	Lossy	Lossless
Bitstream	Scenario A	Scenario B
Data	Scenario C	Scenario D

Notation: In the following, *t* denotes processing time, *E* is the encryption, *SE* is the selective encryption, *C* is compression, *P* is preprocessing for *SE* to identify parts of the data to be encrypted, $>>$ means significantly larger. Attention: *t* is not equivalent to complexity ! In case compression is done in hardware and encryption in software, *t* is significantly smaller for compression, the opposite is true if both stages are executed in software.

Partial Encryption: Scenarios A & B

Data to be encrypted is given as bitstream (or file) B (generated by preceeding compressison). In order to justify selective encryption the following condition must be fulfilled:

$$t(E(B)) >> t(P) + t(SE(B)) \qquad (1)$$

P is the identification of relevant parts of the bitstreams. $t(P)$ can be fairly different: in case of an embedded bitstream or a bitstream with seeral quality layers this effort is almost negligible (the first part of the data or the base layer is encrypted), in a less favourable case we need to partialy decode or at least parse the bitstream to identify important features to be encrypted. In the favourable case, $t(P)$ is negligible, $t(E(B)) >> t(SE(B))$ can be achieed and selective encryption is profitable. In the less favorable case, $t(P)$ might even lead to a flip of the inequality !

Partial Encryption: Scenario C

The raw data I is given (e.g. by previous acquisition) which is compressed in lossy manner. In order to justify selective encryption the following condition must be fulfilled:

$$t(C(I)) + t(E(C(I))) >> t(C(I)) + t(P) + t(SE(C(I))) \qquad (2)$$

P is identical to the scenarios before and all corresponding considerations do apply here. Even in case $t(P) = 0$, inequality (2) can hardly be fulfilled since for most symmetric ciphers and lossy compression schemes we have $t(C(I)) >> t(E(C(I)))$ in case both are executed in software or hardware. Thus, the difference between $t(E(C(I)))$ and $t(SE(C(I)))$ is of minor importance and can often be neglected. For high compression ratios this effect is even more emphasised (since the bitstream to be encrypted is shorter and compression cost is higher in many cases). Therefore, in scenario C selective encryption is not a sensible option.

Partial Encryption: Scenario D

Again, raw data I is given which do not have to be compressed in any case. In case of $t(C(I)) + t(E(C(I))) < t(E(I))$ or compression is required for other reasons (restricted canal capacity), data is compressed in lossless manner and the conditions of scenario C do apply. Since $t(C(I)) >> t(E(I))$ for die most symmetric ciphers and lossless compression schemes, compression is hardly achieved to reduce complexity. Moreover, in the lossless case data reduction is less ignificant which emphasises the contribution of $t(E(C(I)))$.
In order to justify selective encryption (without compression) the following condition must be fulfilled:

$$t(E(I)) >> t(P) + t(SE(I)) \tag{3}$$

P is the identification/extraction of relevant data parts, which can be done in various ways (e.g. wavelets, DCT, bitplanes, ...). In order to satisfy inequality (3) we need to assure $t(P)$ to be small !

EXAMPLE: Encryption of a JPEG2000 image using AES (Scenario C)

The following example (taken from Pommer and Uhl [31]) illustrates the problem of partial encryption in scenario C. We assume that the image is captured, J2K compressed and subsequently encrypted (Online scenario, bitstream oriented encryption). We use the following software:

- J2K: `http://jj2000.epfl.ch`
- AES: `http://www.esat.kuleuven.ac.be/~rijmen/rijndael/`

As example image we use a $512 \cdot 512$ pixel 8 bpp grayvalue image and use as target bitrate 80000 Bit (compression ratio 26). `[]` denotes an array lookup operation, $=$ is an assignment operation, and `^ & + %` denote bitwise exklusive or, bitwise and, Addition and Modulo operation.

AES complexity

name	[ ]	=	^ & + %
KeyAddition	32	16	16
ShiftRow	80	32	32
Substitution	48	16	
MixColumn	136	32	144
128 bit key, 1 block	2858	944	1792
192 bit key, 1 block	3450	1136	2176
256 bit key, 1 block	4042	1328	2304
256 bit key, 80000 bit	2 526 250	830 000	1 440 000

For encrypting 80000 Bit we require about 4 796 250 operations.

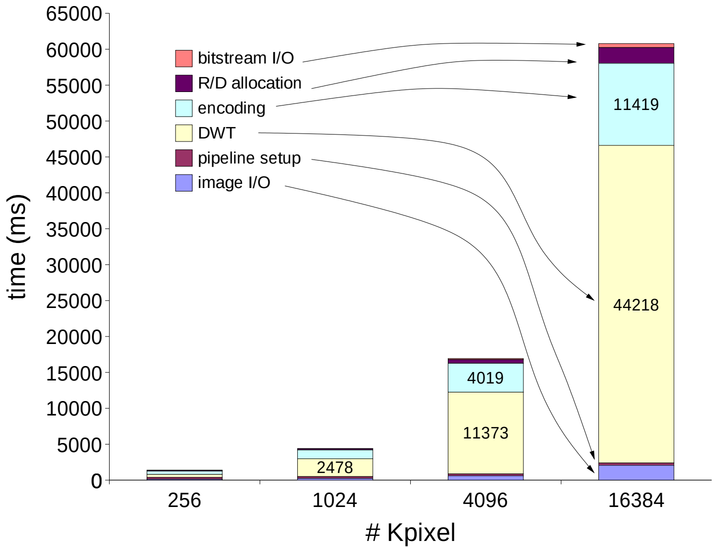
Wavelet Transform Complexity I

image size (length of one side) filter size	N n	level 1 decomposition, 1 line level 1 decomposition, total	$\frac{N}{2} * n$ $2 * \frac{N}{2} * \frac{N}{2} * n = \frac{N^2 n}{2}$
1 line	$N * n$	level i decomposition, 1 line	$\frac{N}{2^i} * n$
1 image (=total)	$2 * N * N * n = 2N^2 n$ (step size = 2, high+lowpass, horizontal+vertical filtering)	level i decomposition, total	$2 * \frac{N}{2^i} * \frac{N}{2^i} * n = \frac{2N^2 n}{2^{2i}}$

Wavelet Transform Complexity II

example values	$N = 512$ pixel side length, $n = 8$
1 operation includes	1 addition $+$, 1 multiplication $\ast$, 2 array lookups `[ ]`
1 decomposition	$2N^2n = 4\,194\,304$ operations
standard wavelet decomposition	$\sum_{i=1}^{5} \frac{2N^2n}{2^{2(i-1)}} = 5\,586\,944$ operations

JPEG2000 Runtime Behaviour



Overall Comparison

In case of using J2K default configuaration (5 decomposition levels, 7/9 biorthogonal filter) we result in 5 586 944 operations for the 512 × 512 pixels image. This number needs to be multiplied by four since each operation involves an addition, a multiplication and two array lookups. Additionally we have at least $\sum_{i=1}^{5} \frac{N^2}{2^{i-1}}$ assignements. This leads in total to 22 855 680 operations for the filtering routine and about 31 744 000 operations for the entire J2K processing chain (when adding 28% processing time for non-filtering related operations).

To compare to: AES required 4 796 250 operations !

JPEG2000 & AES Resume

The operation count for J2K is 7 times higher as compared to AES. Additionally, memory requirements are much higher in the J2K case and the cache behaviour is significantly better for AES (caused by the 128 bit based processing and lots of lookup tabel operations). Wavelet transform on the other hand requires large memory blocks and shows poor cache behaviour due to alternating horizontal vertical filtering.

Testruns using both software implementations showed an AES share below 1% (!!!!) of the entire compression-encryption pipeline.

The immediate consequence is that it makes no sense to decrease the already negligible share of encryption even more by partial encryption since the security is reduced. This is important to keep in mind when assessing the various suggestions made in literature.

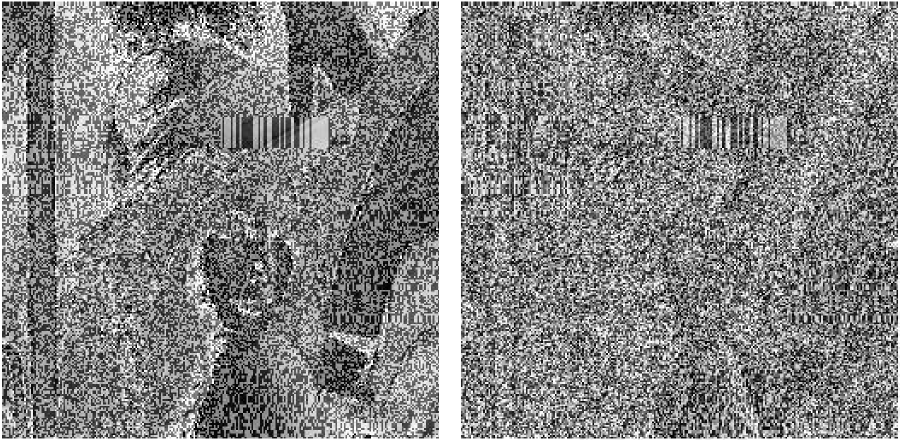
EXAMPLE: Selective Bitplane Encryption (Scenario D)

We assume the aquisition of a digital image, for further transmission or storage no compression is required. In order to apply partial encryption in a sensible way, the identification/extraction of relevant features needs to be fast.

As the fastest technique in this scenario Podesser et al. [29] and Skrepth und Uhl [41] propose selective bitplane encryption. The basic idea is to consider the binary representation of the pixels. A subset of the resulting bitplanes (i.e. binary images at a certain binary position in the pixel representation) is encrypted using AES. The remaining bitplanes are transmitted/stored in plaintext.

In the following we consider 512 $\times$ 512 pixels images with 8 bpp (which results in 8 bitplanes). This leads to a minimal encryption percentage of 12.5 % (1 bitplane). One 128 bit AES block is filled with a quater line ($512/4 = 128$) of a bitplane and processed.

Partial Bitplane Encryption: Examples 1

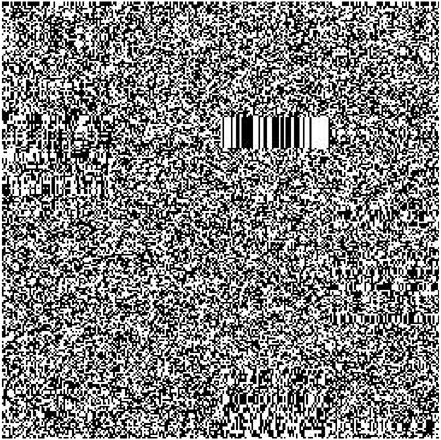


(a) 12.5% encrypted

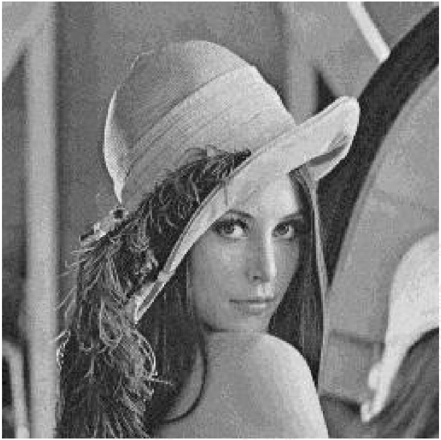
(b) 25% encrypted, 9.0dB

Figure: Visual Examples using direct reconstruction.

Partial Bitplane Encryption: Examples 2



(a) encrypted MSB



(b) 50% encrypted, 31.8dB

Partial Bitplane Encryption: Properties I

- The barcode pattern is generated by encryption of identical neighbouring quater lines using the same key.
- For highest security, the MSB bitplane needs to be encrypted first, the next bitplanes corresponding to follwing positions in the binary representation (see PSNR values in the table).

# Bitplanes	1	2	3	4	5	6	7	8
First: LSB	51	44	38	32	26	20	14	9
First: MSB	9	9	9	9	9	9	9	9

Partial Bitplane Encryption: Properties II

A possibility to increase security would be to secretly choose which bitplanes are subject to encryption in addition to the MSB. This is not a profitable idea for two reasons:

- Encrypted bitplanes close to the LSB do not increase security much, therefore the choice of addiitonal planes is very limited.
- Statistical properties of “natural” and encrypted bitplanes are very different if close to the MSB (and only in this region encryption makes sense). Simple statistical techniques reveal which bitplanes have been subjected to encryption: the table shows the number of runs of 5 identical bits in thousand in the Lena image).

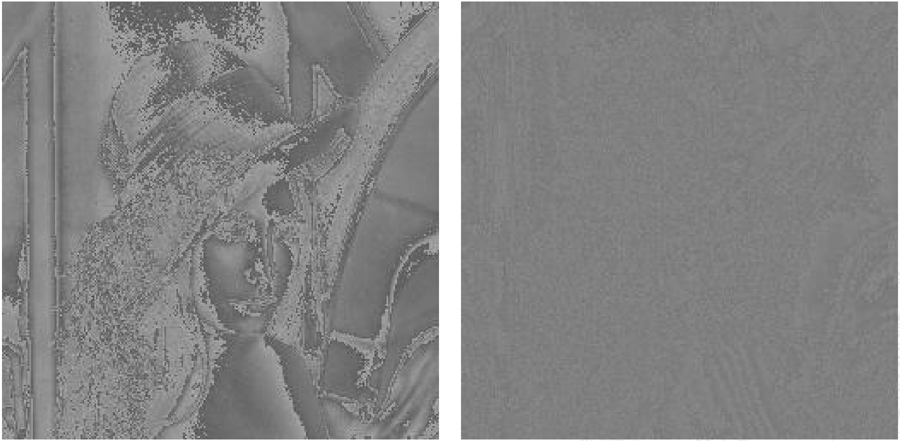
Bitplane	MSB	2	3	4	5	6	7	LSB
Plain	45	39	32	20	11	5	4	4
Encrypted	4	4	4	4	4	4	4	4

Partial Bitplane Encryption: Replacement Attack

In case of direct reconstruction of encrypted material the encrypted parts introduce noise type patterns in the visual data which significantly degrade the image quality. A simple idea is to replace the encrypted parts by “typical” data. In this case we simply introduce a constant 0 bitplane instead of the encrypted bitplane and compensate the loss in average luminance by adding a constant value (depending on the binary position of the replaced plane). We add 64 for the MSB, 96 in case of two planes, and so on ... Reconstruction is then performed as usual.

Whereas in the case of direct reconstruction encrypting 2 bitplanes seemd secure (resulting in 9dB), we recognize important details in the image after the replacement attack (at 13.2dB and severe alienation). Encrypting 4 bitplanes (50% of the original data) resists the replacement attack.

Replacement Attack: Examples



(c) 25% encrypted, 13.2dB

(d) 50% encrypted

Figure: Visual Examples for the Replacement Attack.

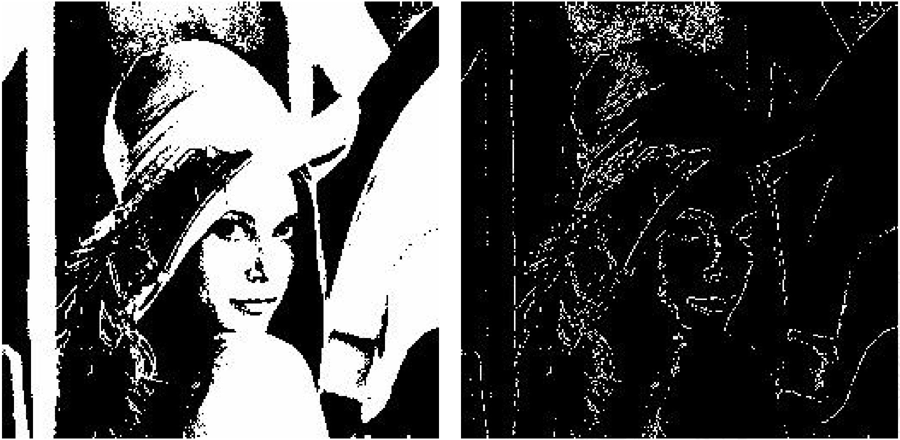
Partial Bitplane Encryption: Reconstruction Attack

The basic idea is to exploit the non-encrypted parts of the data to reconstruct the encrypted parts. We assume only the MSB bitplane to be encrypted and we exploit the smoothness of common images. In smooth regions we expect the MSB values of neighbouring pixels to be identical (except in regions of medium luminance).

In order to identify smooth regions we shift a 2 × 2 pixel search window across the image, where all 16 possible MSB combinations are tested together with the remaining (unencrypted) parts of the data – we compute differences among those 4 pixels. The smallest difference is determined and the corresponding MSB configuration is set as reconstruction.

Edges are clearly marked since a compensation is sought in the window. Setting the MSB in smooth regions to 0 or 1 delivers two half images which may be combined easily to result in a good quality reconstruction. This method gets fairly costly for more encrypted bitplanes but works in principle.

Reconstruction Attack Examples I

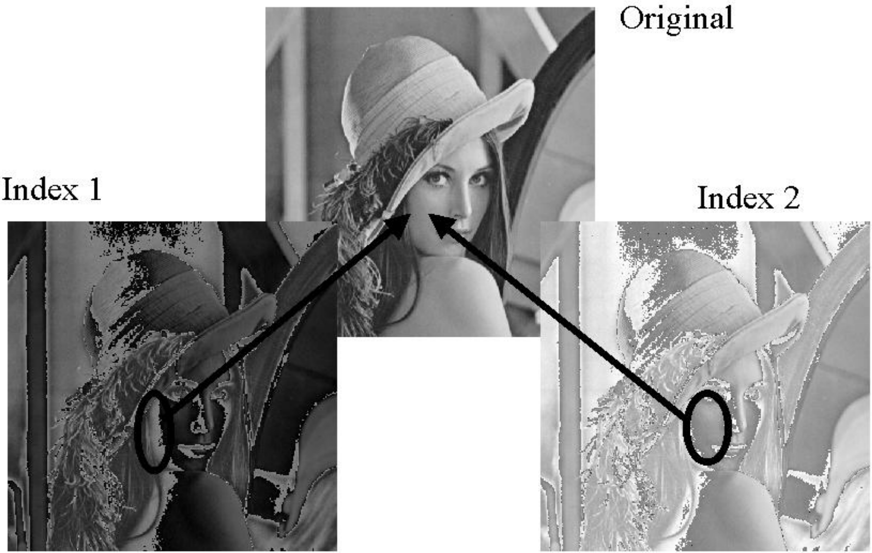


(a) original MSB

(b) reconstructed bitplane

Figure: MSB of Lena and reconstructed MSB bitplane.

Reconstruction Attack Examples II



Compression Integrated Encryption: DCT I

The first algorithm in this class is denoted *Zig-Zag permutation Algorithm* [45, 40]. The main idea is to replace the zig-zag scan of the 8 × 8 DCT coefficients by a secret key-dependent permutation of the coefficients.

- DC-Splitting: the DC coefficient could be identified immediately due to its size. Therefore, before the permutation, the 8 bit DC value is split into two 4 bit blocks, the MSB part remains, the LSB part replaces the last AC coefficient.
- Two additional options increase security:
  - The DC coefficients of 8 blocks are concatenated and DES encrypted, redistributed after encryption.
  - Instead of a fixed permutation list two permutations are employed – the respective use is controlled by a RNG.

Compression Integrated Encryption: DCT I (cont.)

The *Zig-Zag permutation Algorithm* exhibits the following properties:

- PRO: the resulting bitstream conforms to the standard.
- CON: the size of the resulting bitstream is usually significantly larger as compared to the original stream (about 40 - 100 %). The reason is that VLC and RLE are optimized with respect to the zig-zag scan.
- CON: the reduction of complexity is achieved by using a weaker cipher (“soft encryption”) instead of limiting the scope of encryption (“partial encryption”).

Compression Integrated Encryption: DCT I (cont.)

2 Attacks against fixed permutation:

1. Known Plaintext Attack: comparing an original image and an encrypted version immediately reveals the used permutation. The valid permutation is easily identified on a block basis – in the correct case the non-zero AC coefficients tend to gather in the upper left corner of the block. Also, splitting of the DC coefficients alone does not provide enough security since images may be recognised without DC coefficient as well. In [47] this attack is used vice versa (known transform coeffients are fed into a decoder) and is called “Chosen Ciphertext Attack”.
2. Ciphertext Only Attack: Since it is known that the magnitude of AC coefficients usually decreases following the zig-zag scan, a good approximation to the image may be generated testing few configurations only. This may also be done automatically exploiting smoothness constraints (see attacks against Nagravision and Videocrypt).

Compression Integrated Encryption: DCT II

The main problems of the *Zig-Zag permutation Algorithm* result from encrypting data which is not suited for encryption (the properties important for compression are destroyed, the distribution of the coefficients magnitudes is known) and the use of a weak cipher. For improving the scheme, these shortcomings need to be enhanced.

- An alternative solution by Shi und Bhargava [39] suggests to encrypt the sign bits of the DCT coefficients. This does not influence the subsequent compression since these sequences exhibit high entropy anyhow. The first variants of this approach suggest to permute these bitsequences (VEA algorithm, insecure), a latter variant (RVEA) uses DES to encrypt a fixed scan through DC and AC coefficients of luminance and chroma data. Caused by the DPCM encoding of the DC coefficients a significant mixing may be observed. About 10% of the overall data is encrypted. DES may be replaced by AES which results in a very secure scheme. The bitstream remains bitstream compliant.

Compression Integrated Encryption: DCT III

- Zeng und Lei [54] propose to permute coefficients of several 8 × 8 pixels blocks which are positioned at the same (frequency)position in their respective blocks. The result are virtual frequency subbands – this strategy does not suffer from severe compression degradation. For encryption two strategies are suggested:
  1. Permutation using different fixed tables for different virtual bands (which again is not resistant against known plaintext attacks).
  2. Sign Bit Encryption

  Whereas sign bit encryption does not change the size of the bitstream, permutation increases bitstream size by 10%. However, sign bit encryption is not suited for high security requirements since the main image structures are recognisable.

Bitstream based Partial Encryption: Encrypting Headers

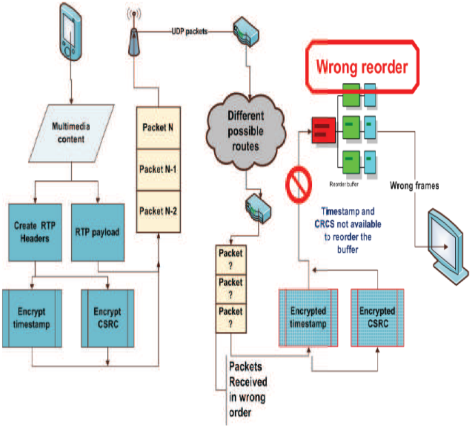
One of the first ideas when encrypting bitstreams is to encrypt header data in order to conceal the data format or to protect parameters required for decompression.

The effort required for encryption is very low – however, it is not sufficient to conceal the image format of course since the underlying format may be identified easily by statistical tests. If the only aim is to block standard viewing tools this approach is the simplest.

In case of protecting header data it is of critical importance not to protect standard information (like image size, bitdepth) - this information can be guessed easily and may be replace by simple cut and paste (from another image). In order to make such a scheme effective, the underlying compression algorithm needs to exhibit several degrees of freedom which may differ from image to image. In case not only main header data is protected, parsing effort for identifying additional header data might get high very soon. Format compliance is lost, but this is the aim of this approach.

Example: RTP Header Encryption

The idea is to encrypt only a subset of the header data of the network-protocol RTP (which is independent of the underlying format, see [2]). Contrasting to this idea is SRTP (RFC 3711) which keeps the header intact but protects payload data only.



Example: Insecurity of RTP Header Encryption

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Idea: It is proposed to encrypt the timestamp and the CSRC field only. The assumption is that caused by the mixing of the packets during transmission and the non-availability of the timestamp, the correct order of the packets can not be reconstructed at the receiving client. However, this is not correct [44]: The sequence number is sufficient to establish the correct ordering (the first packet is assigned a random number, which is successively incremented for the following packets). The CSRC is optional and even in case of encrypting the entire RTP header security is not guaranteed, since the assumption of packet "self permutation" caused by network transmission is not correct in general. Different RTP sessions may be additionally identified by e.g.

Example: JPEG2000 Header Encryption I

Main header encryption usually cannot provide security. An alternative is to encrypt JPEG2000 packet header data selectively. The major motivation is the low amount of header data compared to packet data (the relation depends on the used coding options).

Cblk. Size	Layers	Header Bytes	Body Bytes	Ratio
64×64	16	1823	129072	1.4%
32×32	32	4603	126283	3.5%
16×16	32	11379	119510	8.5%
8×8	32	25748	105176	18.6%

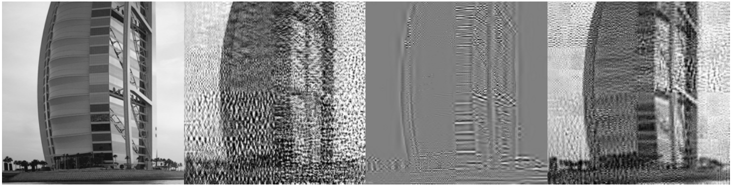
It is obvious, that confidentiality cannot be achieved since actual image data is left unprotected and header data carries the informations how these data have to be interpreted. Target scenario is transparent encryption, where format compliance is a fundamental aspect (in order to be able to view the data using a common decoder). The question is, how this can be acieved using header encryption ?!?

Example: JPEG2000 Header Encryption II

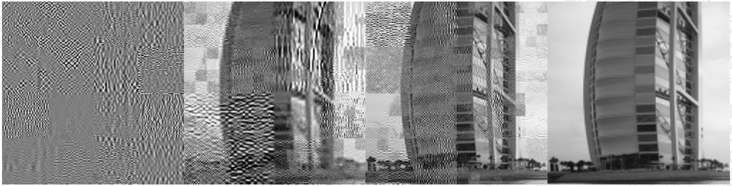
JPEG2000 packet header data consists of inclusion information (i.e. the data of which codeblock are contained in the following packet), the length of the single codeblock contributions (CCP), number of coding passes, and the number of zero bitplanes. A non-availability of these data (or the usage of incorrect data) leads to incorrect interpretation of the packet data during decoding. For transparent encryption, the header data at the beginning of the bitstream (for the pre-view image) remain unprotected.

The required encryption scheme needs to be format compliant: Permutations are used, which are required to maintain certain header properties (e.g. overall header length needs to be preserved).

Example: JPEG2000 Header Encryption Visual Examples



Test image, CCP length and coding passes, number of leading zero bitplanes and inclusion information.



Transformations at resolution 0, 2 and 3, as well as the Preview image

Partial Encryption of Scalable/Embedded Bitstreams

In case the visual data is organized in the form of a base layer and several enhancement layers (JPEG progressive modes, MPEG-2, MPEG-4 scalable profiles, H.264 SVC) encrypting the baselayer [13, 25] is a simple and rather effective possibility to partially encrypt the data. In case of embedded bitstreams (JPEG2000, SPIHT) [6, 19, 28] partial encryption may be adjusted with great accuracy – encryption is done from the start of the bitstream up to an arbitrary truncation point.

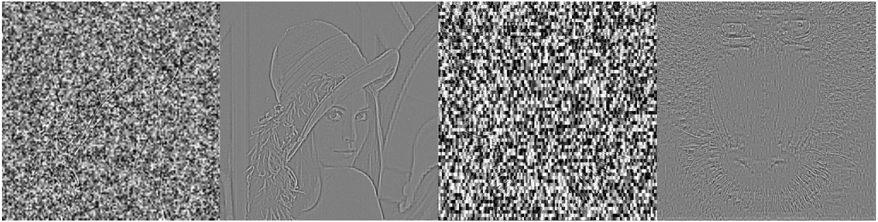
A special application case is *transparent / perceptual encryption*, where a low quality version of the visual material is provided to anybody for free (the base layer [13, 25] or the start of an embedded bitstream [19]) in order to motivate the user to pay for the full quality version. Enhancement layers or the remaining part of the embedded bitstream have been already transmitted to the users with the other parts but in encrypted form. Upon payment of the required fee, the key material for decrypting the encrypted parts is transmitted to the user. The aim is neither complexity reduction nor format compliance but increased functionality as compared to full encryption.

Example: JPEG Extended System I

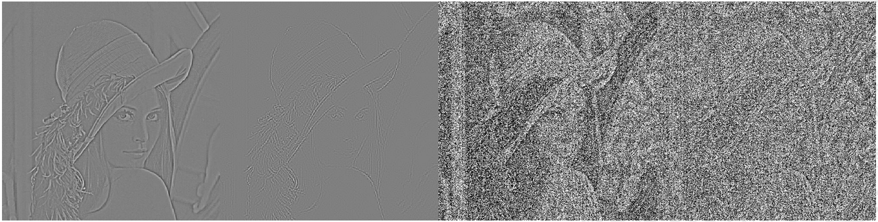
- Hierarchical progressive mode (HP): an image pyramid is constructed by repeated weighted averaging and downsampling. The lowest resolution approximation is stored as JPEG (i.e. the first scan), reconstructed, bilinearly upsampled, and the difference to the next resolution level is computed and stored as JPEG with different quantization strategy.
- Sequential progressive modes
  - Spectral selection (SS): the first scan contains the DC coefficients from each block of the image, subsequent scans may consist of a varying number of AC coefficients, always taking an equal number from each block.
  - Successive approximation (SA): the most significant bits of all coefficients are organized in the first scan, the second scan contains the next bit corresponding to the binary representation of the coefficients, and so on.

The mixed mode interleaves SS and SA.

Example: JPEG Extended System II



3 level HP, lowest level encrypted (direct reconstruction and replacement attack); SS with DC and first AC coefficient encrypted.



HP, SS, SA, MM with 10% encrypted and reconstruction attack (uniform grayscale, zero coefficients, or 0-bitplanes replaced)

Bitstream oriented Partial Encryption: DCT I

The simplest idea is to encrypt the bitstream parts (VLC codewords) which correspond to the leading coefficients in each 8 $\times$ 8 pixels block. The amount of data to be encrypted may be reduced dramatically in this way. However, parsing effort is significant and the remaining data corresponds to a high-pass filtered image which still shows edges and texture information. Although the header structures are maintained, format compliance is usually lost.

Another proposal is to permute the bitstream on a byte level using a fixed permutation table (*Pure permutation Algorithm* [36]). The length of the table may be adjusted to the security requirements and the speed is very high. However, format compliance is of course lost and this technique is again vulnerable by the known plaintext attack.

Bitstream oriented Partial Encryption: DCT II – VEA

A thourough investigation of the statistical properties on a byte basis of DCT-encoded imagery motivates the *Video Encryption Algorithm* [36]. Bytes are considered to be a suitable processing entity:

- Byte operations may be implemented efficiently.
- A single byte has no meaning in the context of a bitstream (VLC codewords consist of more bytes).
- Caused by Huffman VLC codewords bytes show high entropy values.

Bytes in the bitstream turn out to be almost uniformly distributed, the same is true for pairs and triples and arbitrary parts of the bitstream. Additionally, specific byte patterns are hardly ever repeated (which is shown experimentally in [36]).

VEA: Two byte streams are generated (Oddlist – all bytes at an odd position, Evenlist) and Xored. The Evenlist is DES encrypted and concatenated with the previous result. The encrypted evenlist may be seen as a one-time pad.

Bitstream oriented Partial Encryption: DCT II – VEA (cont.)

To enhance security it is proposed to use a key-dependent selection of bytes for the two lists, the selection process is changed for each frame. Addiitonally, the Evenlist is permuted with 8 different permutation lists.

- Security is good.
- Reduction of complexity is 47% only compared to full encryption. However, the scheme can be iterated before actually implementing the encryption of the Even list.
- Non-compliant solution since the structure of the bitstream is entirely destroyed.
- Significant key management effort: DES keys, list generation amd permutation keys.

Bitstream oriented Partial Encryption: DCT III – IPMP

In order to motivate two format compliant encryption schemes for the MPEG-4 Intellectual Property Management and Protection (IPMP) system Wen et al. [52] discuss two specific problems of bitstream encryption.

(1) Even if headers remain intact it is highly probable that markers and headers are emulated if standard encryption is applied which makes a correct interpretation of the bitstream impossible.

(2) The encryption of a concatination of VLC codewords does not lead to a valid concatination of VLC codewords in general. Example: given the codewords 0, 10, 110, 111 and a corresponding concatination 010. The encryption may lead to the result 001 which is not a valid codeword concatination.

Wen et al. [51, 52] propose two algorithms which solve the problems of former techniques:

1. Valid VLC codewords are obtained although strong cryptography is applied (as opposed to e.g. [25]).
2. Security enhancement for VLC codeword permutations.

Bitstream oriented Partial Encryption: DCT III – IPMP (cont.)

The proposed technique for encrypting VLC codewords uses tables with $N = 2^n$ VLC codewords – in case other values are required they are constructed by several smaller 2^n tables.

Before encryption, a n-bit index of fixed length is assigned to each VLC codeword (based on a table). The VLC codewords subject to encryption are concatenated to the string C and the corresponding string S built of the indices is constructed. S is encrypted to S', S' is interpreted via the before-defined table as concatination of VLC codewords C' (S' contains the indices of these codewords). C' is written to the position of C in the bitstream.

It is of course important that the cipher in use does not lead to data expansion (which would lead to a higher number of VLC codewords after encryption). C and C' will have a different number of bits in general. Note that contrasting to VLC, FLC may be encrypted using a strong cipher without problems in general (e.g. DCT sign bits).

Bitstream oriented Partial Encryption: DCT III – IPMP (cont.)

Contrasting to the permutation of coefficients [45, 40] and bytes [36] the aim is to permute semantic entities of the bitstream. Two possibilities are as follows:

1. Entire 8 × 8 pixels blocks are permuted: in this scenario it is important that the number of permuted blocks is large enough to result in sufficient security. A variant is suggested to keep the DC coefficients in place and to encrypt those additionally (these are FLC DPCM encoded). This is similar to permutations in the image domain !
2. VLC codewords are permuted: as already suggested by Zeng und Lei [54], VLC codewords may be grouped to virtual frequency bands to avoid compression degradation. Attention has to be paid to the fact that the number of VLC codewords in the different blocks is different in general.

Bitstream oriented Partial Encryption: DCT III – IPMP (cont.)

In order to increase the security of permutations against known plaintext attacks, an on-the-fly gernation of the permutation tables is suggested. To avoid the high effort required for key management (compare Nagravision or Videocrypt) a generation of the tables using "local" data (i.e. parts of the bitstream not involved in the permutation) is proposed.

In case of VLC codeword permutation DES encrypted DCT sign bits can be used which are not involved in permutation due to their already high entropy.

Bitstream oriented Partial Encryption: DCT III – IPMP (cont.)

Example for generating a permutation table:

- DCT sign bits und DC information is DES encrypted (key K_F).
- A random bit sequence R_L of length L is generated using a RNG and the key K_L (fixed for a frame). $L > bitlength \times K$ for all $bitlength \times K$, where K is the number of codewords in a table and *bitlength* is $log_2(K)$.
- For each set of codewords to be permuted the encrypted sign bits are concatenated to R'.
- R' is encrypted using the key K_T which results in the output R, which is repeated *bitlength* times, which results in Rr.
- $Rc = R_L \quad XOR \quad Rr$.
- Rc is cut into K non overlapping segments, each with *bitlength* bits. The permutation table maps each index input value i (from 0 to $K - 1$) to the i-th segment of Rc.

Bitstream oriented Partial Encryption: DCT III – IPMP Evaluation

- Partial encryption may be employed for maintaining format compliance but also for reducing complexity (if the number of VLC codewords protected is limited corresponding to other proposals).
- Permutations are made resistant against known plaintext attacks without introducing a huge key mamgement cost. Vulnerability against ciphertext only attacks (distribution of AC coefficients magnitudes is known) remains.
- The processing overhead (especially for encryption of VLC codewords via index tables) is significant, this is also true for extracting sign bit sequences for constructing the permutation tables.
- Currently, these proposals are among the best for DCT imagery, however, there remain still limitations as we have seen.

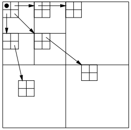
Compression integrated Image Encryption: Wavelets I

Similar to the DCT case two approaches are discussed to manipulate coefficients, the assessment of which is slightly different:

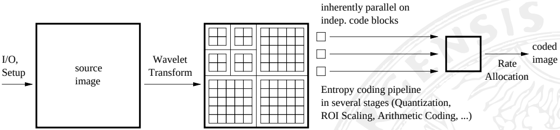
1. Encryption of coefficient sign bits [54, 19]: similar to the DCT case sign bits of the wavelet transform coefficients exhibit high entropy and are therefore good candidates for encryption since they cannot be compressed anyway. Due to the spatial localisation of the wavelet coefficients edges remain visible to some extent. Format compliance is maintained.
2. Permutation of coefficients [54, 48]: of course permutations are vulnerable by the known plaintext attack also in this case without countermeasures taken. [48] suggests additional encryption of the low pass subband to increase security (which leads to loss of format compliance on the other hand). [54] combines permutation with block rotation and sign bit encryption. Contrasting to the DCT case a ciphertext only attack is not possible due to the image dependent localsation of the coefficients. Also, the compression degradation is lower as compared to the DCT case, however, is gets more severe the more advanced the wavelet coding scheme is. Context based algorithms (like SPIHT and EZW) have been shown to suffer more from

Example: Permutations in the Wavelet Domain I

Whereas **S**et **P**artitioning **I**n **H**ierarchical **T**rees relies on spatial orientation trees (S.O.T. – zerotree like structures) thereby using inter **and** intra subband correlations, JPEG2000 coding is based on codeblocks which limits its scope to intra subband correlation.



S.O.T.



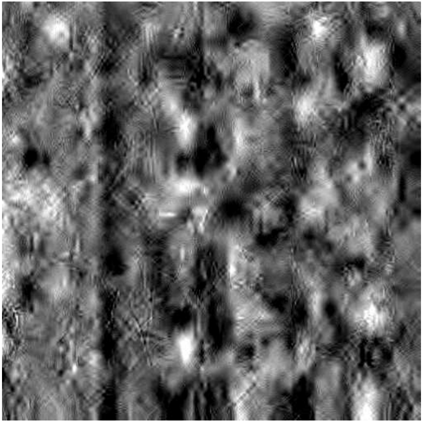
J2K Coding pipeline

Example: Permutations in the Wavelet Domain II

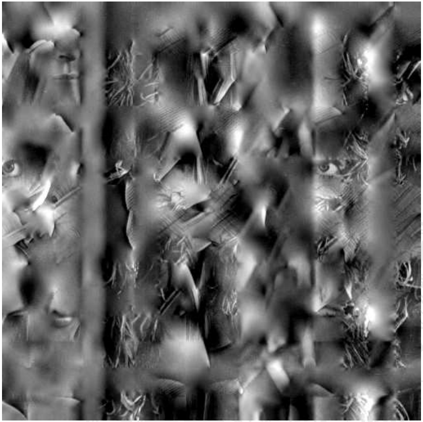
The following permutation scenarios are considered:

1. **blockwise-fixed**: Blocks of fixed size across all subbands are used ($1 \times 1, 2 \times 2, 4 \times 4, 8 \times 8,$ or 16×16 coefficients).
2. **blockwise-adaptive**: Each wavelet-subband is divided into 64 equally sized blocks.
   - arbitrary permutations across the subbands
   - identical permutations across all subbands: here entire multiscale trees are permuted as a whole !!

Example: Permutations in the Wavelet Domain III



Arbitrary Permutations



Identical Permutations

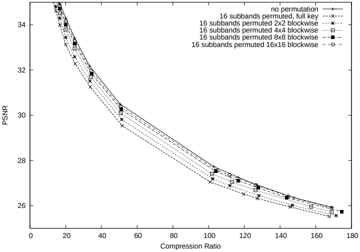
Example: Permutations in the Wavelet Domain IV

Each testimage is encoded with both considered coding algorithms. Within the coding pipeline, the coefficients of the different wavelet subbands are permuted before the quantization stage using one of the proposed permutation variants. The filesize (i.e. compression ratio) is recorded.

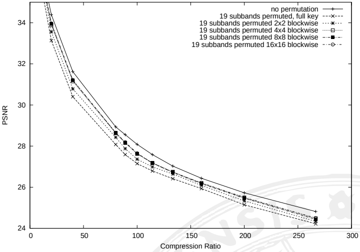
Thereafter, the encrypted and compressed file is decoded and the corresponding wavelet-subbands inversely permuted ('decrypted'). The image quality is recorded. Finally the overall rate versus distortion performance is computed.

Additionally, the efficiency loss measured in percentage of filesize increase as compared to the original coder is computed.

Example: Permutations in the Wavelet Domain V



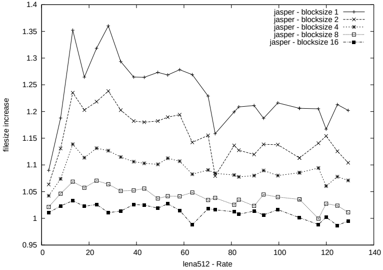
JPEG2000, blockwise-fixed



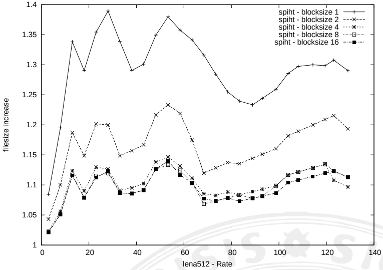
SPIHT, blockwise-fixed

In this representation, differences are hardly visible !

Example: Permutations in the Wavelet Domain VI



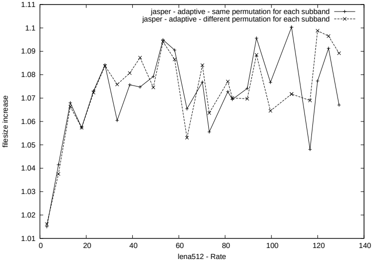
lena512, JPEG2000, blockwise-fixed



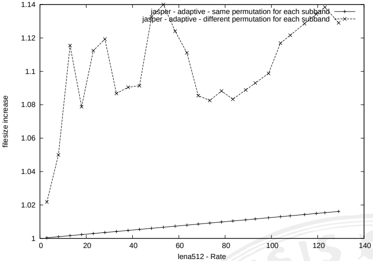
lena512, SPIHT, blockwise-fixe

There are subtle differences between the results of JPEG2000 and SPIHT. In the JPEG2000 case increasing the blocksize improves the result steadily up to blocksize 16 × 16, but we notice a saturation of the improvement at a blocksize of 4 × 4 pixels for SPIHT. Contrasting to JPEG2000, SPIHT cannot take advantage of a larger blocksize.

Example: Permutations in the Wavelet Domain VII



JPEG2000



SPIHT

We clearly note the different behaviour of SPIHT vs. JPEG2000: in the SPIHT case, using identical permutations accross the subbands does not harm the compression efficiency whereas it does not make a difference for JPEG2000. The inter subband correlations as preserved contribute about 10% of the file size in the SPIHT case.

Compression integrated Image Encryption: Key-dependent Wavelet Transforms

Wavelet transformation may be performed in a large variety of ways – as a consequence it is possible to develop a special type of header encryption scheme which takes advantage of “secret transform domains”. The main idea is that it is sufficient to protect the exact type of transform as defined in the header of the compressed file. Minimal compression effort and partial bitstream compliance may be achieved (recall that the data parts of the bitstream remain in plaintext).

- Wavelet Packets [32, 33, 34] employ secret subband structures.
- NSMRA: use of different (secret) wavelet filters at different levels of the decomposition [30].
- Parameterized wavelets: wavelet filters are chosen out of a huge parameter space [10, 35].

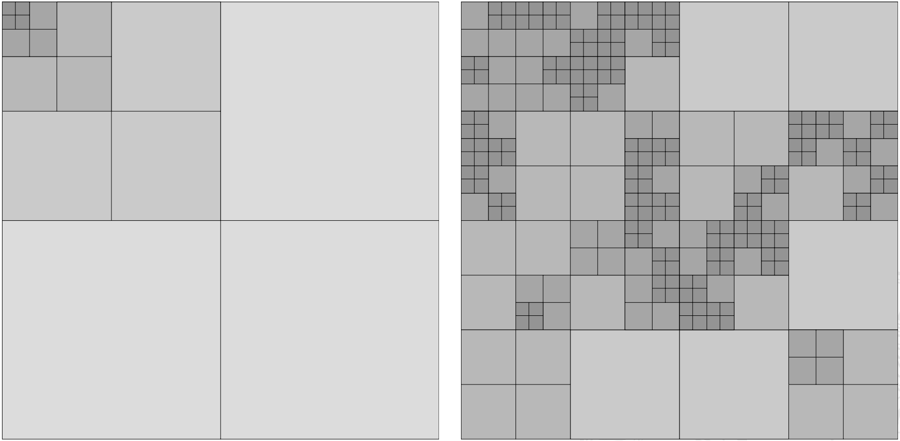
Main issues to be investigated when using these techniques are the control of compression quality and reconstruction of the kind of transformation used when analyzing transform coefficients.

Key-dependent Wavelet Packets: Basics I

Wavelet Packets gerneralize the pyramidal wavelet transform and do not only decompose recursively the low-pass subband but all subbands are subject to recursive decomposition. The consequence is a better frequency resolution, in particular with respect to higher frequencies. In the 2-D case we result in a full quadtree, out of which those subbands are chosen which represent the data in a desired way. Application areas for wavelet packets:

- Compression: best basis algorithms with cost functions, FBI fingerprint standard, J2K Part 2
- Signal classification, especially for textured signals
- Numerics for representing operators in a compact way.

Key-dependent Wavelet Packets: Basics II



(a) Standard wavelet decomposition

(b) Typical wavelet packet decomposition

Figure: Decomposition trees for different subband structures

Key-dependent Wavelet Packets: Basics III

Main idea: a RNG generates a quadtree structure which defines the wavelet packet quadtree structure. The image is subsequently decomposed following the quadtree and compressed. Only the seed of the RNG needs to be stored in encrypted manner to reconstruct the tree structure at the decoder, the remaining data (i.e. coefficients) remain unencrypted.

The generation of the quadtree structure is based on a random decicion for each subband (decompose or not). A subband structure with depth 1 has probability $1/2$ (image decomposed or not), a structure with depth 5 only $\frac{1}{2^5}$. Therefore, shallow decomposition have a much higher probability than deep ones - this is corrected using weights in the generation of the tree.

An important question is the compression quality of those randomly generated structures and if some structures need to be excluded due to their low quality. It turns out that a certain minimal decomposition depth for the low pass subband needs to be assured.

Key-dependent Wavelet Packets: Quality

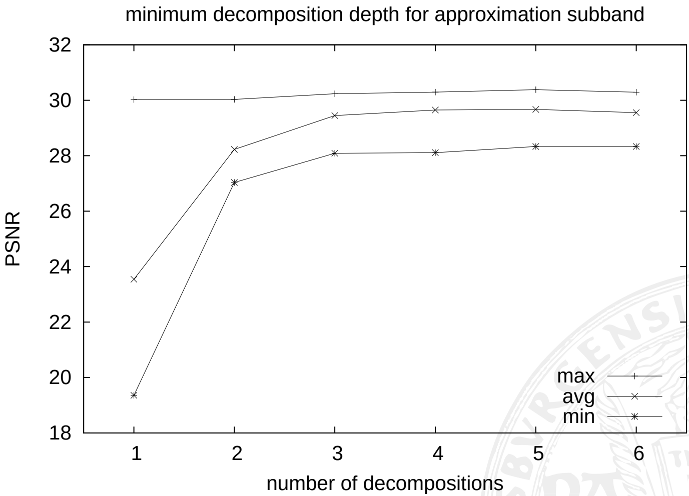


Figure: approximation subband minimum decomposition depth

Key-dependent Wavelet Packets: Security

Possible attacks against the scheme:

- Attack the encryption of the RNG seed: unrealistic (e.g. AES encryption)
- Brute force attack against the seed: good RNG does not enable that attack
- Reconstruction of the subband structure based on coefficient analysis: discussed in the following
- Exhaustive search through subband structures: when restricting the decomposition depth to 6 (which also makes sense for compression quality) results in 2^{4185} different subband structures

Key-dependent Wavelet Packets: Attacks I

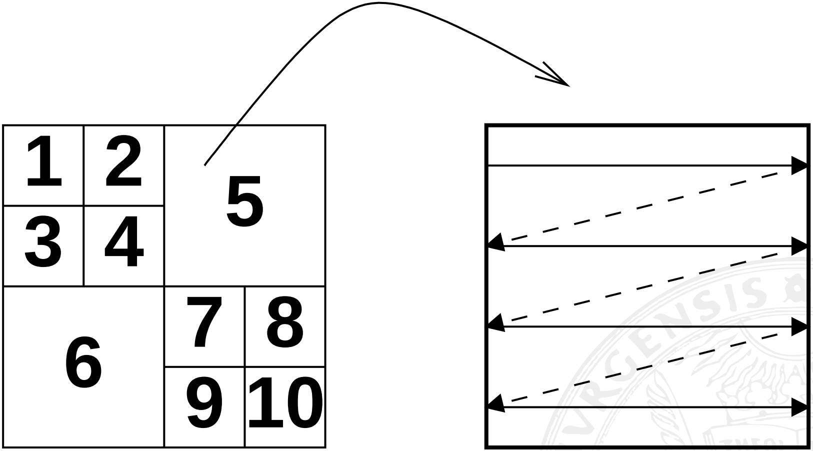


Figure: Example for our scan order assumption

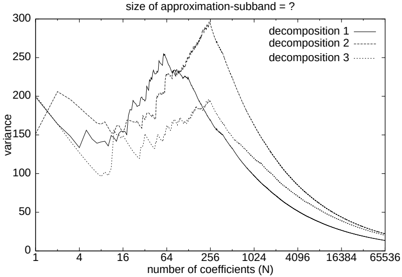
Key-dependent Wavelet Packets: Attacks II

The first and most important stage in an attack to reconstruct the subband structure is to identify the coefficients of the low pass subband. Statistical properties of these coefficients significantly differ from those of the other subbands (i.e. close to uniform vs. Laplace distribution with high number of zeros, respectively).

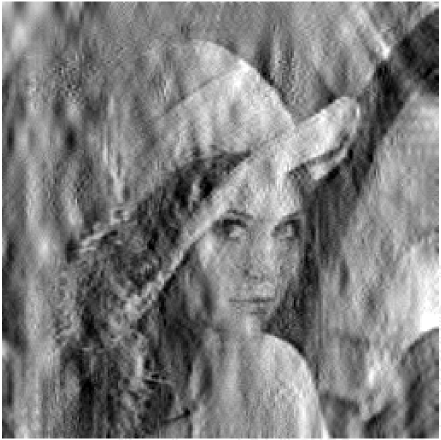
We can exploit this knowledge by computing a modified variance $v = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - m)^2$, m being the coefficients' mean and N the number of considered coefficients. When evaluating v for an increasing number of coefficients, we notice a clear decrease of the values at some particular value, which indicates the end of the low pass subband.

The size of the detail subbands can be found by correlating lines with appropriate subband dimansions, where high correlations indicate the use of a correct size. Note that this attack cannot be used in case of zero-tree based codecs, since coefficients are neither stored according to their original subband nor as entire coefficients.

Key-dependent Wavelet Packets: Attacks III



(a) Variance for increasing number of coefficients



(b) Reconstruction using a wrong decomposition tree but the correct approximation subband size

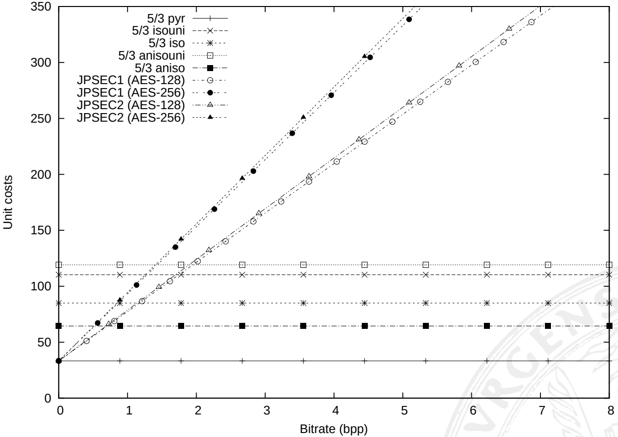
Key-dependent Wavelet Packets: Efficiency I

The main motivation for the entire approach is the negligible amount of data to be encrypted (i.e. the subband structure) – in the following we discuss the overall complexity for the following reference systems [15]:

- JPSEC1: Traditional AES Encryption of a JPEG2000 file
- JPSEC2: Format compliant AES Encryption of the JPEG2000 file (i.e. packet header preservation, no marker emulation)
- isouni: WP-based encryption without considering quality
- iso: WP-based encryption when considering quality

For estimating the complexity of the WP-based schemes, the average decomposition depth needs to be determined which is generated by random decompositions.

Key-dependent Wavelet Packets: Efficiency II



Key-dependent Wavelet Packets: Efficiency III

Complexity estaimation is based on an idealised model with unit costs for operations and memory access. Actual run-time:

structure	compression		decompression	
pyramidal	0.643 s	1.55 fps	0.404 s	2.47 fps
iso	1.005 s	1 fps	0.962 s	1.04 fps
isouni	1.147 s	0.87 fps	1.011 s	0.99 fps

JPSEC encryption			
Method	throughput	codestreams with 2bpp	
JPSEC1	42.71 MB/s	0.0015 s	683.36 fps
JPSEC2	37.81 MB/s	0.0017 s	604.96 fps

Thus, overall, compression time dominates by far, and consequently, the significnat run-time increase caused by WPs cannot be compensated by a negligible encryption extent !!

Remark: This is true only for for current hard- and software configurations (e.g. codecs being hardly optimised while AES encryption software is highly tuned, poor cache behaviour of the

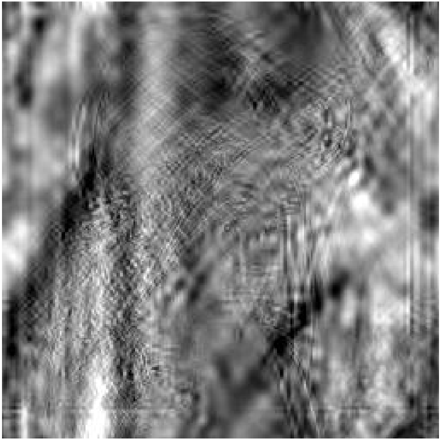
Key-dependent NSMRA Decomposition

Non-stationary Multiresolution Analyses (NSMRA) are pyramidal wavelet decompositions which employ different filters at different levels of the decomposition. The basic idea is similar to the wavelet packet case: only the information which filter is used at which decomposition level needs to be encrypted. For the technique described in [30] a filter library consisting of l filters and a decomposition depth of k is used. This results in a key space of l^k, the filter combination used for compression is randomly chosen (again, only the seed needs to be encrypted).

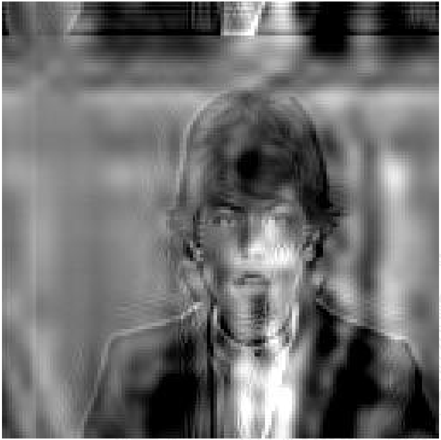
A brute force attack is not feasible provided the filter library is large enough – however, there exists an attack against the scheme which reduces the number of filters required to be investigated to $l \times k$: is the correct filter used at a certain decomposition level the resulting subband exhibits a lower amount of noise as compared to the case where the wrong filter was used. We use the entropy of the difference of neighbouring pixels as measure of smoothness in our attack. The attack proceeds level by level, fixing at each level the best filter found (the filter leading to coefficients with lowest entropy values).

The existence of this attack makes the scheme rather insecure.

Key-dependent NSMRA Decomposition: Example



(c) Reconstruction using (wrong) random filters



(d) Reconstructed image where the heuristic failed at 3 out of 5 levels

Key-dependent Wavelet Filter Parametrisations I

Basic idea is to select wavelet filters from a large familiy of parameterised filters in a key-dependent manner and to employ these filters for compression. Only the parameter(s) need to be encrypted. Problems may include:

- Filters generated by different parameters will exhibit varying quality w.r.t. compression effectiveness and it is not clear a priori how to steer key-selection in a quality-aware manner. It is also not clear if enough good-quality filters exist for providing a sensibly large key-space.
- Filters with similar parameter values behave similar in terms of compression, which reduces key-space and facilitates attacks (gradient descent !).
- Bi-orthogonal parametrisations exhibit high fluctuations w.r.t. compression quality, while they tend to compress better than orthogonal ones (compare bi-orthogonal 9/7 filter).

Key-dependent Wavelet Filter Parametrisations II

Dilation equation for orthogonal wavelets with compact support:

$$\phi(t) = \sum_{k \in Z} c_k \phi(2t - k),$$

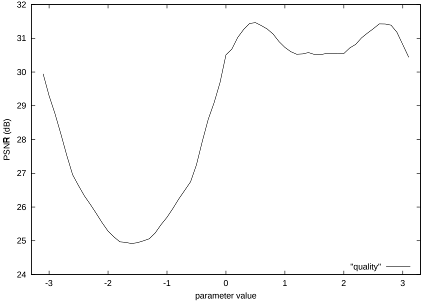
with $c_k \in R$, and several conditions to be fulfilled by the coefficients c_k. The most well known parametrisation has been proposed by Schneid und Pittner [38]. We assume N parameters $-\pi \leq \alpha_i < \pi$, $0 \leq i < N$, to be given (which should serve a key material), then the follwoing recursion can be used to determine the coefficients c_k^N, $0 \leq k < 2N + 2$:

$$c_0^0 \;=\; \tfrac{1}{\sqrt{2}} \quad \text{and} \quad c_1^0 \;=\; \tfrac{1}{\sqrt{2}}$$

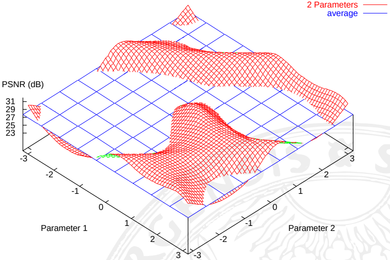
$$c_k^n \;=\; \tfrac{1}{2}\Big((c_{k-2}^{n-1} + c_k^{n-1}) \cdot (1 + \cos \alpha_{n-1}) + (c_{2(n+1)-k-1}^{n-1} - c_{2(n+1)-k-3}^{n-1})$$

We set $c_k = 0$ für $k < 0$ and $k \geq 2N + 2$.

Key-dependent Wavelet Filter Parametrisations: Quality

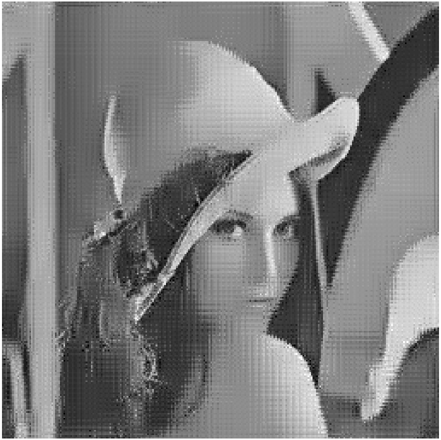


(e) Quality Range using Parameterised Filters

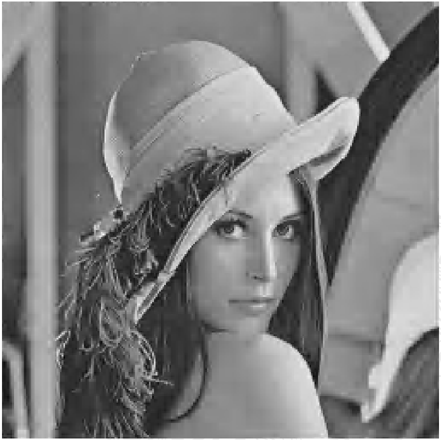


(f) Quality, using a 2-dimensional parameter space and average

Key-dependent Wavelet Filter Parametrisations: Quality Examples



(g) Best Quality



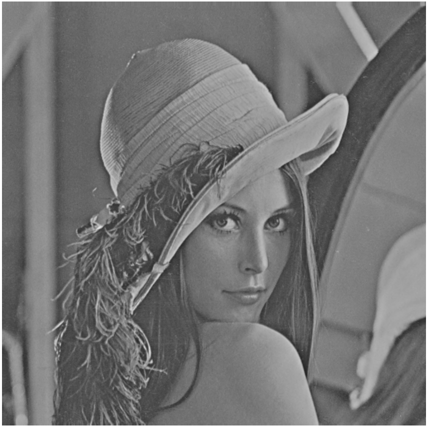
(h) Worst Quality

Key-dependent Wavelet Filter Parametrisations: Security

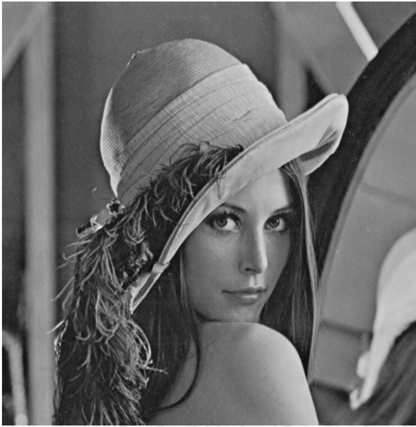
Using the parameterised representation of the coefficients, it is possible to compute a symbolic inverse wavelet transform (the encrypted image is transformed back to spatial domain). As a result, at each pixel position we have a symbolic expression using the parameters of the filter coefficients [14].

- Ciphertext-only Attack: A function exhibiting high correlation with the pixel values needs to be optimised (minimised), e.g. sum of pixel differences or variance. The type of employed parametrisation determins if this works.
- Known Plaintext Attack: The potential availability of a pre-view image facilitates the estimation of single pixel values (in larger homogeneous areas) or the average value in an area could be estimated (only 1 parameter can be estimated). The type of parametrisation determines the number of required pixels to conduct the attack.

Key-dependent Wavelet Filter Parametrisations: Security Examples

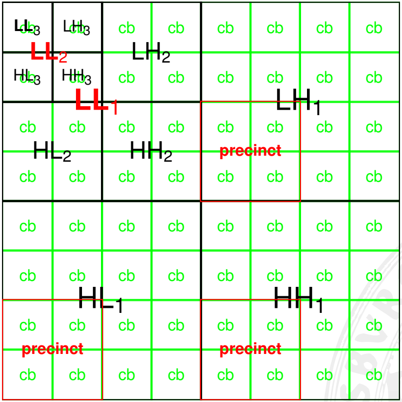


Ciphertext-only, Variance, 1 Par.



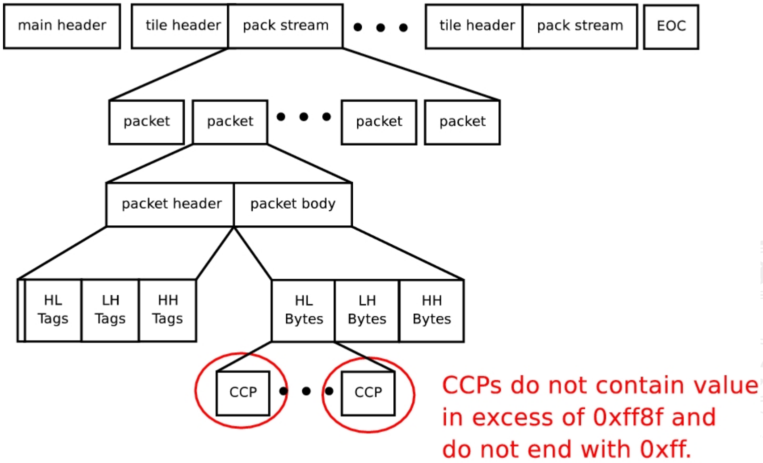
Known-Plaintext, Avg. Illumination, 1

Bitstream-oriented JPEG2000 Encryption: Bitstream Structure



Data layout of the JPEG2000 transform domain with code blocks (cb) and larger structures.

Bitstream-oriented JPEG2000 Encryption: Compliance I



CCPs do not contain value in excess of 0xff8f and do not end with 0xff.

The syntax of the code stream does not allow a 2 byte sequence in

Bitstream-oriented JPEG2000 Encryption: Compliance II

For all CCPs:

1. Encrypt the CCP.
2. Check if it contains a two byte sequence in excess of 0xff8f.
   If yes goto 1 and reencrypt the encrypted CCP.
3. Check if it ends with 0xff.
   If yes goto 1 and reencrypt the encrypted CCP.
4. Output the code-stream compliant encrypted CCP.

Data encrypted in that manner can be decrypted in trivial manner: Decode until compliance is re-gained, since the original data was compliant as well. To result in an efficient scheme, coding options have to be set such that CCP are not too long (otherwise too many iterations are required) [43].

Bitstream-oriented JPEG2000 Encryption: JPSEC

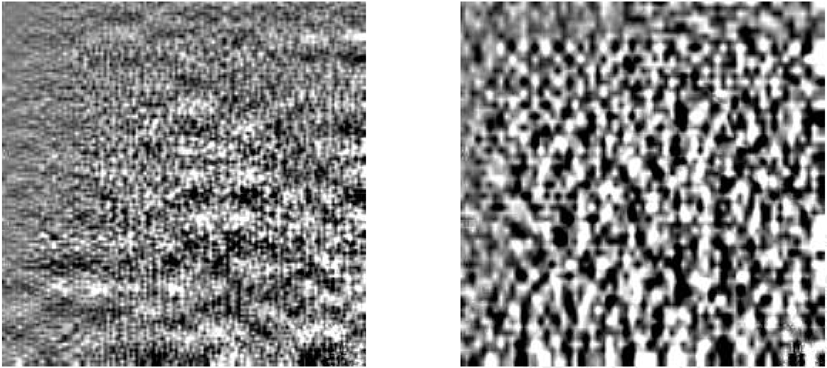
Due to the embeddedness of the J2K bitstream partial encryption of the bitstream can be achieved easily by restricting the encryption to the first part of the bitstream. When doing this without caution format compliance is destroyed.

For this reason [19, 28] suggest to encrypt packet data only – all header structures remain intact. However, a significant encryption reduction is required to make the parsing for packet data profitable !

The implementation in Norcen et al. [28] uses JJ2000 and resolution and layer progressive modes are compared for their respective suitedness for this application, Note that marker emulation is an important topic for format compliance here !

An important class of attack is the replacement attack [28] (also called error concealment attack by Wen et al. [52]) where the encrypted parts of the bitstream are replaced or ignored while decoding. Error concealment techniques implemented in J2K may be used here (if special marker are set the decoder ignores non compliant stream parts).

Bitstream-oriented JPEG2000 Encryption: JPSEC Examples I

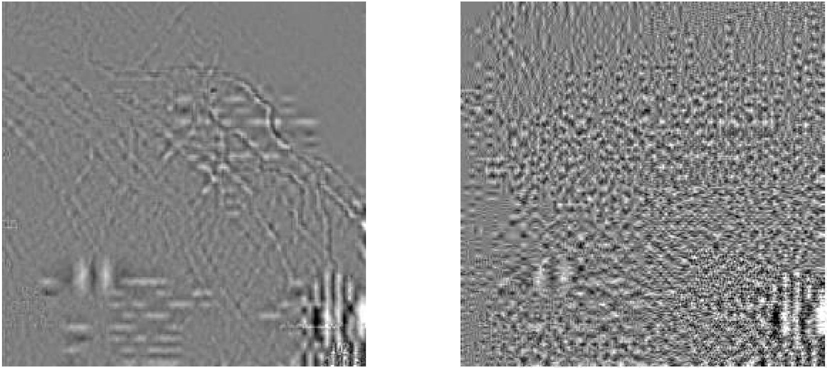


(i) layer progressive, 8.79 dB

(j) resolution progressive, 7.45 dB

Figure: Comparison of selective encryption (visual quality of reconstructed Angio2 where 1% of the bitstream data have been encrypted) using resolution or layer progressive encoding.

Bitstream-oriented JPEG2000 Encryption: JPSEC Examples II



(a) 1% encrypted, 10.51 dB

(b) 20% encrypted, 9.90 dB

Figure: Visual quality of reconstructed Angio2 after replacement attack using resolution encoding.

Bitstream Oriented Partial Encryption: SPIHT

Set Partitioning in Hierarchical Trees (SPIHT) is an advanced version of zerotree coding replacing lists by sets. During compression, three different sets are maintained: sets of insignificant pixels (LIS), insignificant isolated pixels (LIP) and significant pixels (LSP). These sets describe zerotree structures. When generating the bitstream, sign bits, refinement bits and significance bits of pixels and set of pixels are generated.

A decoder needs to interpret bits in the correct context (i.e. the correct txpe) to generate valid results. In particular, incorrect significance bits lead to a misinterpretation of subsequent bits, contrasting to incorrect sign and refinement bits. This observation is exploited by [6] and encrypts significance bits of low resolution subbands. As a result, the synchronisation between encoder and decoder is lost. The amount of encrypted data is very small, when taking care even format compliance may be preserved.

Partial Image Encryption using Quadtrees I

Quadtree compression is a very fast non-standardised procedure providing better rate-distortion performance at low bitrates as compared to JPEG. Can be implemented bottom-up or top-down.

1. Lossless Mode: Knots in the tree degernerate to leaves in case they share a common gray value and do not need to be decomposed further. The tree structure and the gray values at the leaves are being stored (identical number of bits for all leaves).
2. Lossy Mode: Instead of testing for a common gray scale potential leaves are tested for similarity (e.g. pixel variance, similar average gray scale, etc.). The number of bits to represent leaves-values varies (therefore we result in lossy compression, depending on the size of the leaves, the variance of leave values at this quad-tree level, ...).

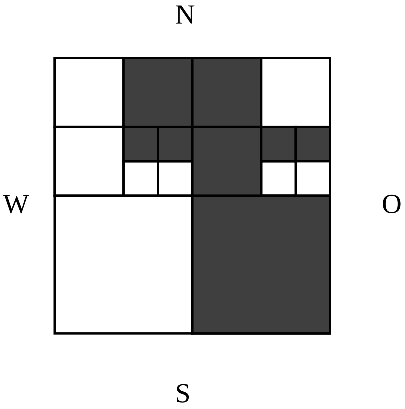
In the lossless case, 14% is an upper bound for the share of quadtree information w.r.t. the overall data volume for 8bpp imagery, typical values for the lossy case are 15 – 25 % across a wide range of bitrates.

Partial Image Encryption using Quadtrees II

Cheng et al. [5, 6] propose to encrypt the quadtree leaving the leaf values in plaintext. Similar to the wavelet packet case a brute force attack is not feasible due to the high number of possible quadtrees. By analogy, one can try to derive the quadtree structure by analysing the plaintext data. In this context, two possible variants of organising the leave values are discussed:

1. Leaf Ordering I: is a Depth First ordering starting with the NW quadrant, always counter-clockwise.
2. Leaf Ordering II: is a line-orneted scan of all leaves of one quadtree level, starting with the smallest leaves.

Partial Image Encryption using Quadtrees III



Leaf Ordering I: 0010011011110010

Leaf Ordering II: 1111000001100101

Figure: Example for leaf ordering I and II.

Partial Image Encryption using Quadtrees IV

It turns out that there is an attack against leaf ordering I, while no attack is known against leaf ordering II. The basis of the known attack is that in leaf ordering I the leave values of neighbouring leaves are neighbours in the bitstream as well. Based on this fact it turns out that runs (i.e. a sequence of identical leaf values) provide information about subtrees, which drastically reduces the number of admissble quadtrees.

In particular, we exploit that four identical leaf values cannot belong to four leaves at the same quadtree level (if they did, no partitioning had been taken place). In addition to this knowledge, homogeneous regions can be reconstructed.

Partial Encryption: Videos

All techniques discussed so far may be applied straightforward to I-frames (and in principle also to P- and B-frames).
Additionally, video specific properties and structures may be exploited for partial encryption (these approaches may be combined of course):

- GOP structure: the difference of I-, P-, and B-frames with respect to information content can be exploited to implement partial encryption. This technique has been entitled “selective encryption” originally.
- Motion vectors: since motion vectors describe how to build predictions of P- and B-frames based on I-frames they are more important as the compressed error frame. Motion vectors may be encrypted as well of course.

Selective Video Encryption I

Maples and Spanos [42] propose to restrict MPEG encryption to the encryption of I-frames. Since I-frames are the basis for predicting P- and B-frames, the video stream should be sufficiently protected.

Agi and Gong [1] soon showed the insecurity of the approach caused by the I-blocks present in P- and B-frames – too much visual information remains in the video. As a consequence, all I-blocks need to be encrypted at least (as proposed by Wu et al. [53] in combination with a watermarking approach).

Selective Video Encryption II

Meyer and Gadegast [36] developed 1993 - 1995 SECMPEG which further refines the above approach. SECMPEG uses DES, RSA, and CRC (for checking integrity) and a dedicated file format has been developed which is based on MPEG but is not compatible to MPEG. Therefore, dedicated encoder and decoder units are required. SECMPEG provides four security levels:

1. Header encryption
2. Header, DC and first AC coefficients of I-frames are encrypted
3. All I-blocks are encrypted
4. Full encryption

Since the motion vectors are not protected object and camera motion remains visible in this approach.

Partial Video Encryption: Motion Vectors I

If motion vectors are encrypted in addition to I-frames the problem of I-blocks in P- and B-frames remains. MV protection has to be seen therefore as an additional security measure only – when setting all MVs to 0 we usually obtain a good frame approximation (error concealment attack).

- Zeng and Lei [54] propose an encryption of the MV sign bits and a permutation of MVs. This leads to data expansion (but only moderately since the MV share is already small).
- Wen et al. [52] propose to encrypt VLC MV codewords again based on encrypting an index list. Again we face slight data expansion since the original codewords contain many 0s (short codes).
- Shi et al. [39] propose to protect sign bits of MVs in addition to protecting sign bits of DCT coefficients sign bits. Their scheme encrypts 64 bits per macroblock.

Partial Video Encryption: Motion Vectors II

Cheng and Li [6] propose tree structures for the encryption of MVs. The idea is to represent the motion vector field as quadtree, the leaf values are motion vectors instead of gray values in that case. The quadtree structure is encrypted, leaf values remain in plaintext.

A problem of this approach is that small video formats lead to small quadtrees (only 1 MV is stored per macroblock), which requires a full encryption to provide sufficient security. Using leaf ordering II is of special importance here since the MV field contains many homogeneous regions.

This technique is better suited to protect more detailed MV fields (e.g. as required for wavelet SVC).

Special Media Encryption Techniques

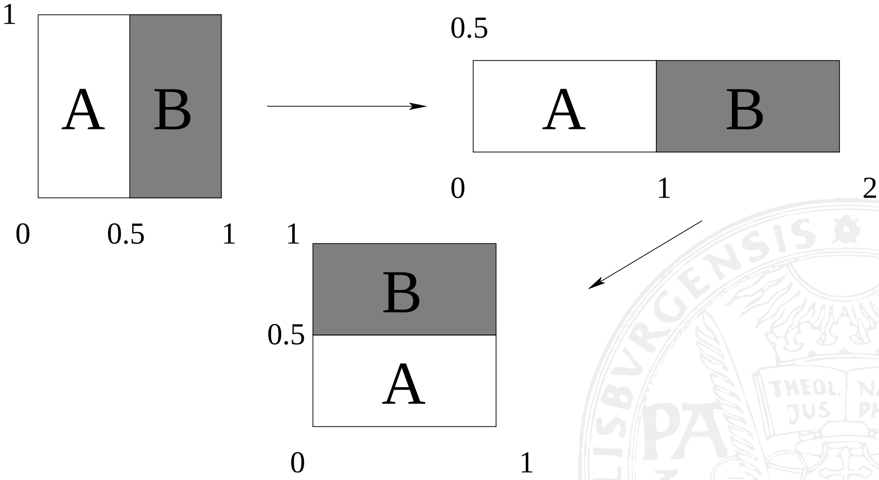
The techniques discussed so far either employ very fast cipher to adapt to visual content or restrict the encryption to specifically chosen parts of the data. In the following we discuss two techniques which may be employed for both, still images and videos.

- Image encryption using chaotic maps: a class of chaotic maps exhibiting strong mixing properties is used for encrypting visual data (“chaotic mixing”).
- Error robust encryption (for wireless channels): transmitting encrypted material over channels with transmission errors results in a large amount of plaintext destroyed (due to the mixing properties of high quality ciphers). This is of course not desired for error-prone channels or storage media.

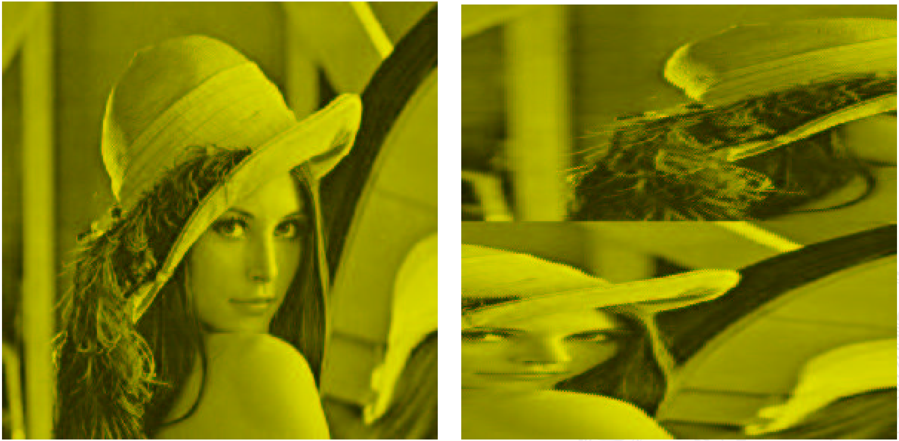
Chaotic Mixing I

This class of encryption techniques [37] is based on the fact that chaotic maps exhibit properties similar to those shown by encryption algorithms. The main difference is that chaotic maps are not discrete “by nature”. A discretization of chaotic maps is the basis of all related techniques. The best know chaotic map is the “Baker Map” B defined on $[0,1]^2$:
$B(x,y) = (2x, y/2)$ for $0 \leq x < 1/2$ and $B(x,y) = (2x-1, y/2+1/2)$ for $1/2 \leq x \leq 1$.
The left vertical half $[0,1/2) \times [0,1)$ is streched horizontally and scaled vertically and is mapped to $[0,1) \times [0,1/2)$. Similarly, the right half $[1/2,1) \times [0,1)$ is mapped to $[0,1) \times [1/2,1)$.

Chaotic Mixing II



Chaotic Mixing III

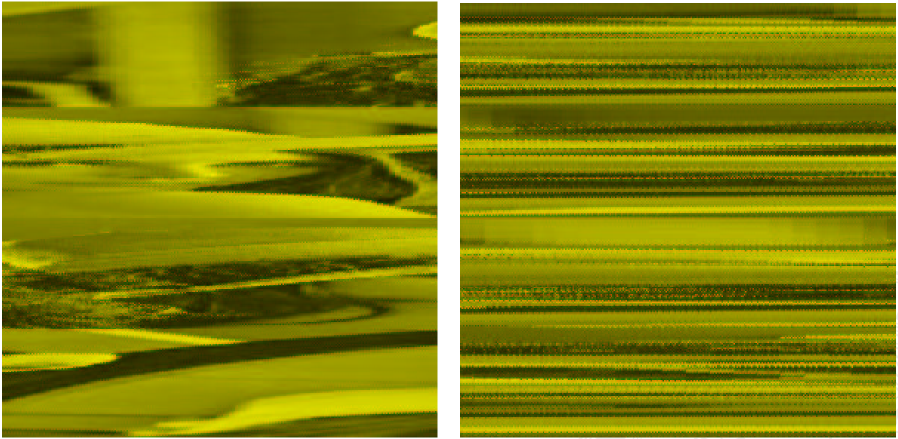


(a) Lena (yellow ;-)

(b) 1 Iteration

Figure: Baker Map (1/2,1/2).

Chaotic Mixing IV

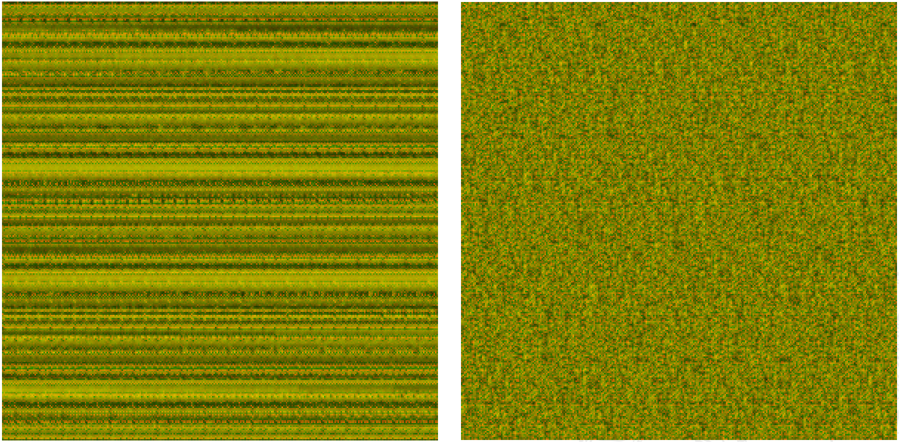


(a) 2 Iterationen

(b) 4 Iterationen

Figure: Baker Map (1/2,1/2).

Chaotic Mixing V



(a) 6 Iterationen

(b) 10 Iterationen

Figure: Baker Map (1/2,1/2).

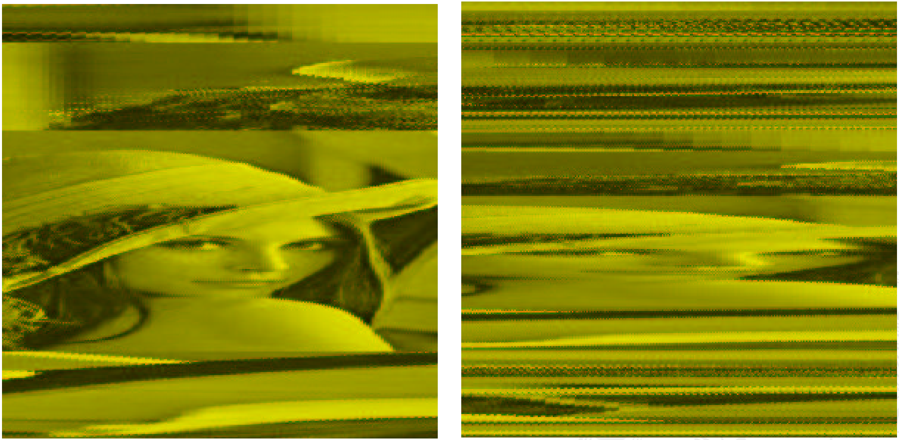
Chaotic Mixing VI

The Baker Map may be generalized as follows: instead of using two halves we partition the unit square into k vertical rectangles $[F_{i-1}, F_i) \times [0, 1)$ for $i = 1, \ldots, k$ and $F_i = p_1 + p_2 + \cdots + p_i$, $F_0 = 0$, such that $p_1 + \ldots p_k = 1$. The left lower corner of the i-th rectangle is F_i. This generalization streches each rectangle horizontally by the factor $1/p_i$ and scales it vertically by p_i. Finally the rectangles are stapled above each other.

For applying this map to an image with pixels it needs to be discretized; the resulting map needs to be a bijective map among pixels. We define a sequence of k numbers $n_1, \ldots, n_k$ such that each n_i divides the image width N of a square image without remainder and $n_1 + \cdots + n_k = N$. $N_i = n_1 + \cdots + n_i$ and $N_0 = 0$. The pixel (r, s) with $N_{i-1} \leq r < N_i$ and $0 \leq s < N$ is mapped to (with $q_i = N/n_i$):

$$(q_i(r - N_i) + s \pmod q)_i) \,, (s - s \pmod q)_i)/q_i + N_i) \,.$$

Chaotic Mixing VII



(a) 1 Iteration

(b) 2 Iterationen

Figure: Baker Map (1/10,1/5,1/2,1/5).

Chaotic Mixing VIII

The algorithm discussed so far is a pure pixel permutation defined by the discretized Baker Map. The permutation key is the choice of the n_i. In order to achieve a mixing of the gray values the map is further generalized to three dimensions. A pixel (r, s) with gray value g_{rs} is mapped to $B(r, s)$ with gray value $h(r, s, g_{rs})$ which means that the new gray value depends on pixel position and former gray value. To ensure invertability, h has to be a bijection in the third variable, e.g. $h(r, s, g_{rs}) = g_{rs} + r * s \pmod{L}$ where L is the number of gray values. Now this algorithm is a mix of permutation and substitution and leads to a uniformly distributed histogram after few iterations. In order to add diffusion a simple non-linear RNG is applied collumn after collumn ($g^*_{rs} = g_{rs} + G(g^*_{rs-1}) \pmod{L}$ with arbitrary seed value). The key of the entire scheme consists of the parameters of the chaotic map, the iteration number, the parameters of the gray value transformation h and the parameter of the RNG.

Chaotic Mixing IX

Summarizing, we obtain the following properties:

- The scheme operates in the spatial domain without involving compression. However, it could be applied as well to coefficients in some transform domain.
- High memory demand since the transformations operate on the entire frame (which is bad for hardware implementations).
- High speed due to simplicity of the operations involved.
- Security of chaos based encryption is discussed controversially, depends on the quality of the discretization.
- For small images the keyspace is small.
- Robust against transmission errors and compression

Error Robust Encryption I

The problem of classical encryption in error-prone environments is that ciphertext errors expand into larger errors in the plaintext. The reason are cipher properties important for the security:

- Similar plaintexts lead to entirly different ciphertexts (important to withstand known plaintext attacks)
- Avalanche effect: changing one input bit changes 50% of the output bits
- Completeness: each ciphertext bits depends on all plaintext bits

In error prone environments massive FEC (forward error correction) is required to protect encrypted material. The question is if we can do better for these applications ?

Error Robust Encryption II

Tosun und Feng [46] conduct a systematic investigation which encryption schemes E do not exhibit these undesired properties. The following property is desired for wireless environments: if E(x) and E(y) differ by c bits, x and y differ by d $\leq$ c bits.

A class of encryption schemes showing this property are “error preserving” algorithms for which holds $d(x, y) = d(E(x), E(y))$ where $d(x, y)$ is the Hamming distance. In their work the authors characterize such algorithms.

It turns out that all error preserving algorithms may be expressed by bitpermutation and bitcomplement ! This shows that employing permutations for encryption of visual data does not only make sense due to the high speed but also from the error robustness viewpoint.

Permutation changes on a per frame basis and table generation on the fly make this approach an intersting candidate for multimedia data.

Encryption of 3D Graphics Data: Notions

- A mesh M is a tuple (V, K).
- V is a set of vertex positions v_i with $v_i := (x, y, z)$; $x, y, z \in \mathbb{R}$ (geometry information)
- K is a set of faces of the mesh with a face $k_j := (e_l, e_m, e_n)$ being defined as a triplet of edges of M (connectivity information)
- e_k is a tuple of vertices of M. $e_k := (v_i, v_j)$; $v_i, v_j \in V$
- $Nb(v_i)$ are the (unprotected / un-encrypted) direct neighbours of v_i. $Nb(v_i) := \{v_j \mid (v_i, v_j) \in M\}$
- $INb(v_i)$ are the (unprotected / un-encrypted) indirect neighbours of v_i. Indirect neighbours are connected to v_i via a vertex positioned in-between the neighbour and v_i.
  $INb(v_i) := \{v_j \mid \exists v_k \in M : (v_j, v_k) \in M \wedge (v_k, v_i) \in M \wedge v_k \notin Nb(v_i)\}$

Encryption of 3D Graphics Data: Simple Techniques

While in literature many examples for watermarking of 3D data can be found (“second line of defense”), not much work is done in the area of encryption (“second line of defense”):

- A first approach allows the viewer of a client application to load the graphics only in low resolution. The client sends the required viewing angle to the server which renders an image with high-resolution data and sends it to the client. In this manner, the client never gets access to the 3D graphics data itself in high resolution. However, protected transmission of transparent encryption cannot be facilitated using this approach.
- A second approach applies partial encryption to a randomly selected subset of vertex and connectivity informations. This approach is problematic, since on the one hand, using a low amount of encrypted data security is very low, on the other hand, for decent security, a large extent of the data has to be protected. The reason for this fact is the high redundancy of the data.

Basic idea in [21, 20]: Employment of a hierarchical mesh-representation by analogy to scalable or embedded data formats.

Encryption of 3D Graphics Data: Hierarchical Meshes

1. Mesh Simplification: Simple strategies to reduce data amount to facilitate e.g. fast rendering.
   - Feature Removal: Iteratives removal of small detail features
   - Vertex Clustering: Several vertices are replaced by a single one
2. Progressive Meshes: Less detailed mesh versions can be visualised with a lower amount of data and for the full resolution, data of the lower resolution are re-used.
   - Compressing progressive Meshes: Based on the low resolution data, a prediction is computed for the missing vertex data of the higher resolution and only the error of this prediction is stored. The aim is best-possible data reduction.
   - Non-compressing progressive Meshes: Additional data corresponding to the higher resolution is stored directly. The aim is a lower computational effort despite the usage of a progressive format as compared to compressing progressive meshes. Of course, the resulting representation is less compact.

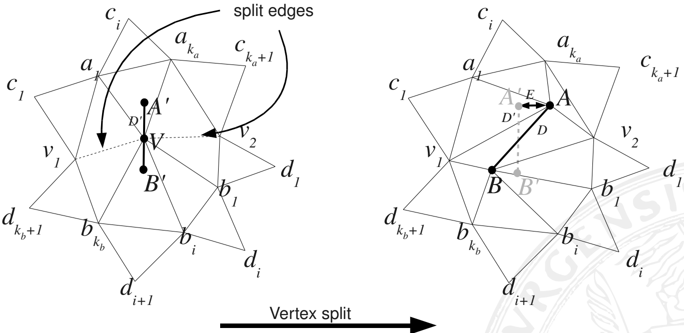
Encryption of 3D Graphics Data: Vertex Split / Collapse I

How can we generate lower resolution meshes and mesh predictions for higher resolutions ?

1. Two vertices A & B are selected the removal of which has the lowest impact to the mesh (edge collapse).
2. Those two are collapsed into vertex V, V is marked as a split-candidate. Vertices (A', B') are estimated / predicted based on their neighbourhood (vertex-split of V) (e.g. a_k, b_k, c_k, v_1 and v_2 are used to predict A')
3. Vector $D = A - B$ represents the distance bewteen A und B, only the error vector $E = D - D'$ (with $D' = A' - B'$) is being stored.

Subsequently, only vertices are collapsed which have not been used in the prediction of (A', B'). All possible corresponding vertices constitute a quality layer, a repeated application of the procedure leads to several quality layers.

Encryption of 3D Graphics Data: Vertex Split / Collapse II

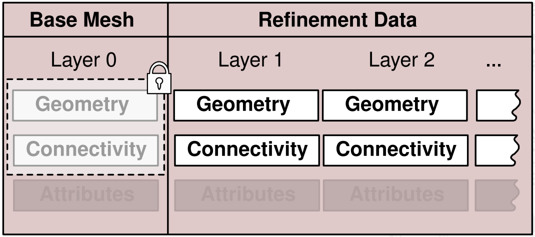


(a)

(b)

Encryption of 3D Graphics Data: Encryption of Non-compressive Progressive Meshes

The first encryption variant considers the encryption of non-compressing progressive meshes, geometry information (i.e. vertices) is encrypted, we start from the begin of the geometry data. The un-encrypted part of the data can be decoded / rendered without any hassle.



Encryption of 3D Graphics Data: Attacks

As it is the case with visual data, a direct decoding of the entire data introduces noise into the data. The simplest attack sets all encrypted data to $(0.0, 0.0, 0.0)$ a more advanced idea sets the value to the mean of all vertices in plaintext. Based on connectivity information an attacker is able to identify unencrypted vertices closed to the encrypted ones and can therefore compute a linear interpolation for v_i: $P(v_i)$.

$$Avg_1(v_i) = \sum_{v_k \in Nb(v_i)} \frac{v_k}{\#Nb(v_i)}, \quad Avg_2(v_i) = \sum_{v_k \in INb(v_i)} \frac{v_k}{\#INb(v_i)}$$

$$\text{if} \#Nb(v_i) > 1 : \quad P(v_i) = Avg_1(v_i)$$

$$\text{otherwise} : \quad P(v_i) = \frac{2 * \#Nb(v_i) * Avg_1(v_i) + \#INb(v_i) * Avg_2(v_i)}{2 * \#Nb(v_i) + \#INb(v_i)}$$

The result can be interpreted in a similar manner as the interpolation of an encrypted pixel in an image. With an increasing amount of vertices being encrypted, the quality of the prediction decreases.

Encryption of 3D Graphics Data: Error Measures

How can we measure errors in 3D graphics data ? Contrasting to visual metrics which are often computed pixel-wise, a vertex-wise error cannot be computed since in two meshes to be compared the are not necessarily corresponding vertices in existence. Thus, a variant of the Hausdorff distance is often used: The procedure uses randomly selected points on the first mesh M, for each of those points p the minimal distance to the second mesh M' is computed. The maximum taken over these distances gives the Hausdorff distance.

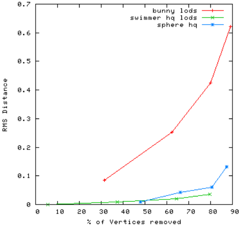
$$d(p, M') = \min_{p' \in M'} \|p - p'\|_2$$

$$d(M, M') = \max_{p \in M} d(p, M')$$

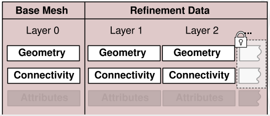
Since in this first variant statistical outlyers are over-emphasised, the average taken over all points of mesh M is computed:

$$d_{rmse}(M, M') = \sqrt{\frac{1}{|M|} \sum d(p, M')^2}$$

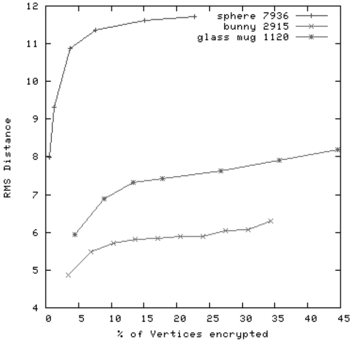
Encryption of 3D Graphics Data: Error Measure Examples

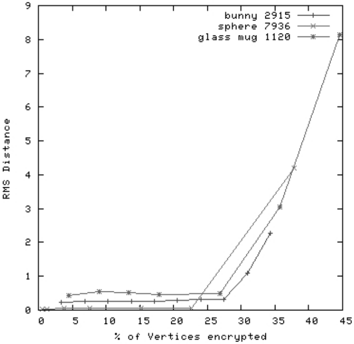


Obviously, the measure works as desired. The more vertices are omitted, the larger the error / distance gets. This corresponds to an encryption of an increasing amount of refinement data (which is the simplest approach, comparable to transparent encryption of scalable media formats).



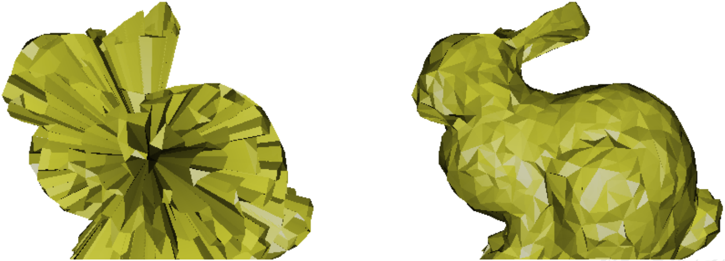
Encryption of 3D Graphics Data: Comparing Attack Variants I





Distance when encrypting (starting at the "top" of the file) and attacking a non-compressing mesh – the attack technique employing the average midpoint leads to significant errors (left), however, the attack using neighbourhood data provides low errors for up to 30% of data encrypted (right).

Encryption of 3D Graphics Data: Comparing Attack Variants II



Obviously, the attack variant employing local averages instead of simple $(0, 0, 0)$ leads to much better results, as correctly predicted by the error measure (the leading 500 vertices are encrypted and replaced by averaged values).

Encryption of 3D Graphics Data: Encryption of Compressing Progressive Meshes

Compressed progressive meshes can be encrypted in two variants: base mesh encryption and encryption of the first refinement layer. In compressing progressive meshes, positions of vertices v_i of the higher quality are being predicted and only the prediction error E_{v_i} is stored.

$$E_{v_i} = v_i - P(v_1, v_2, .., v_{i-1}, v_{i+1}, .., v_n)$$

Considering the second variant (refinement layer encryption), all vertices v_i' decoded from encrypted data (prediction based on the encrypted data plus stored errors) are based on incorrect predictions and are therefore error-prone:

$$v_i' = P(v_1', v_2', .., v_{i-1}', v_{i+1}', .., v_n') + E_{v_i}$$

When inserting E_{v_i} into this equation we get:

$$v_i' = v_i - P(v_1, v_2, .., v_{i-1}, v_{i+1}, .., v_n) + P(v_1', v_2', .., v_{i-1}', v_{i+1}', .., v_n')$$

Obviously, the predictor itself plays an important role when reconstructing the data.

Encryption of 3D Graphics Data: Predictor Types

- Linear prediction: for $v_j \in Nb(v_i)$, $a_j = 1$, otherwise 0:

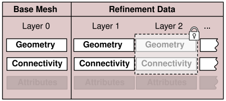
$$P_L(v_1, .., v_{i-1}, v_{i+1}, .., v_n) := \frac{a_1 v_1 + .. + a_n v_n}{\# Nb(v_i)}$$

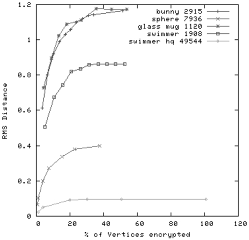
- Butterfly prediction: for $v_j \in Nb(v_i)$, $a_j = 1$, else 0 and for $v_j \in INb(v_i)$, $b_j = 1$, else 0 ($\alpha = 1.15$ is typical):

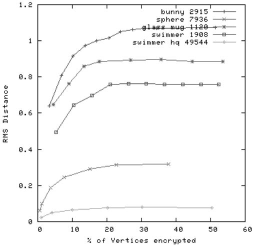
$$P_L(v_1, .., v_{i-1}, v_{i+1}, .., v_n) :=$$

$$\alpha * \frac{a_1 v_1 + .. + a_n v_n}{\# Nb(v_i)} + (1 - \alpha) * \frac{b_1 v_1 + .. + b_n v_n}{\# INb(v_i)}$$

Encryption of 3D Graphics Data: Reconstruction Examples



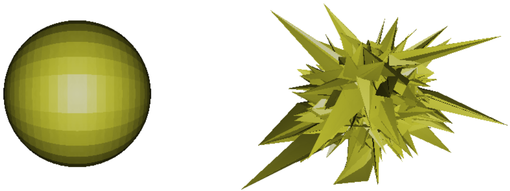




The difference between the predictors is clearly visible. It gets also very clear that this encryption strategy can only be employed for transparent encryption since it is too weak for confidentiality purposes or even sufficient encryption.

Encryption of 3D Graphics Data: Base Layer Encryption

For content confidentiality base layer encryption of compressing progressive meshes seems to be appropriate:



A major difference to image and video data is that it is not clear which “typical” data to use for the base mesh in a replacement attack. Also a reconstruction attack (eventually using mesh-smoothness as criterium) seems to be difficult.

Assessment of (Visual) Encryption Schemes: Security Notions

In order to be able to assess the security of such encryption schemes, it needs to be clarified which type of security we are talking about:

- MP security (message privacy): Information about the ciphertext must not lead to informations about the plaintext. E.g. here we cannot encrypt compressed media data since file size as such delivers information about the plaintext. Difficult to achieve.
- MR security (message recovery): The aim is an entire reconstruction of the plaintext. Encryption of the back-parts of a scalable bitstream has identical MR-security as encryption of the entire bitstream. Not very useful.
- MQ security (message quality): The aim of an attack is a good *approximative* reconstruction of the data with high perceptual quality.

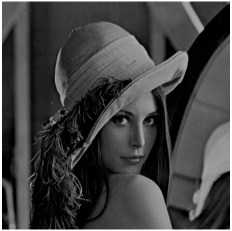
The latter notion of security includes a notion of quality, this is why we have a look at this issue.

Assessment of (Visual) Encryption Schemes: Metrics

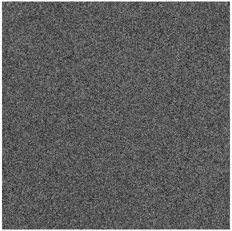
Image / video quality metrics have been typically developed for mid- to high quality data, the few exceptions developed for far are not really convincing:

- Mean opinion score (MOS): Averaged quality assessment by human observers. All numerical metrics should exibit high correlation to MOS.
- PSNR: Pixel-wise squared error; does not even work well for high quality, highly questionable for encrypted data.
- Structural similarity index (SSIM): Combined similarity w.r.t. luminance, contrast and structure into a final score. Currently the top choice for mid- to high quality data in terms of correlation to MOS.
- Luminance similarity score (LSS): In 8×8 pixel blocks changes in luminance are summed up in weighted manner.
- Edge similarity score (ESS): In 8×8 pixel blocks dominant edge directions are compared.
- Local entropy: On a local basis, the difference to a random image w.r.t. entropy is computed.

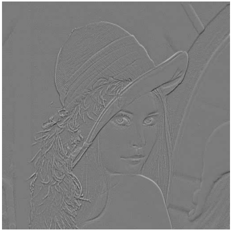
Assessment of (Visual) Encryption Schemes: Metrics Examples



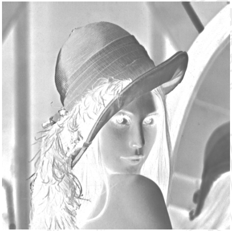
Original



10.8dB, LSS-1.4,ESS 0.17, SSIM 0.02



10.6dB, LSS-1.4,ESS 0.17, SSIM 0.49



4.1dB, LSS-4.6,ESS 1.0, SSIM 0.02

Assessment of (Visual) Encryption Schemes: Lessons Learnt

Currently available metrics exhibit low correlation to MOS / perception for (partially) encrypted data. It has to be noted that the notion of quality is only relevant for transparent and sufficient encryption, for content security it is more the question of intelligibility and recognition of image content that actually matters. This has to be assessed in different manner.
As a possible example biometrics can be used: After applying a privacy-preserving encryption scheme for surveillance video data and attacks against it, we conduct automated face recognition to the encrypted and attacked data to investigate eventual automated recognition. The same can be done for encrypted fingerprints or iris textures, where recognition schemes can be applied to partially encrypted image data.

A potential strategy for “general purpose” image data: In CBIR various methods to determine similarity of images and objects in images exist, often based on SIFT, SURF and similar descriptors. These techniques can also be applied to encrypted data and compared to the results of un-encrypted data. This is a topic of current research.

Outline

1. Formalia
2. Introduction & Motivation
3. Encryption of Visual Data
   - Basics
   - Pay TV Systems
   - DVD Security
   - BLU RAY Security
   - Partial Encryption Schemes
4. Information Hiding & Watermarking
   - Robust Watermarking
   - Steganography
   - Integrity Verification for Visual Data
5. Literature

Information Hiding & Watermarking

Digital media offer the possibility to embed additional data (the *payload*) into into the main data, i.e. the *cover data* (contrasting to embedding data into meta data of a specific format): “Information Hiding” [23, 22]. For this generic idea, different application areas exist, i.e.

- DRM / Robust Watermarking: Originating from the need to protect intellectual property and copyright , “Digital Watermarking” [7, 12] has been introduced, embedding imperceptible copyright information into the cover data in a way that it cannot be removed without destroying the value of the cover data. Termed as “Second line of defence” in digital rights management, once encryption has been broken.
- Steganography (“hidden writing”): Developed as an alternative to classical crypto – in case encryption cannot be applied, another option to provide confidential communication is to conceal the fact that communication takes place.
- (Semi-)Fragile Watermarking: A common problem is unauthorised tampering of digital media data without leaving perceptual traces. Providing means to protect integrity of digital media data is therefore an important complement to classical crypto techniques like hash functions.

Information Hiding & Watermarking: Notions

In DRM oriented information hiding, we distinguish *watermarking*, where all copies of the data are being treated equally (e.g. copyright or copy control information), and *fingerprinting*, where different data are embedded into different copies of the data (e.g. to *trace traitors*, i.e. to identify a subject / item, that has not followed license regulation, e.g. by spreading video data across networks in illegitimate manner after having acquired it).

Classical *steganography* is very different, also in terms what an efficient attack is considered to be: Proving that data have been embedded is considered a successful attack in most stego application scenarios. Contrasting to that, in DRM information hiding, involved parties are mostly aware of the fact that they operate with marked data, thus, a successful attack requires to destroy or remove the embedded data.

Further application areas of *robust* watermarking are annotation WMs (meta informations, URLs), copy control WMs (e.g. DVD CGMS), and broadcast monotoring (e.g. quality control – strength of embedded data decreases with decreasing cover data quality – or broadcast protocols).

Information Hiding & Watermarking: Requirements / Properties I

Information Hiding requirements can be quite different and do often contradict each other. For specific application scenarios (e.g. robust WM, stego, fragile WM), different requirements / properties are emphasized.

- Robustness: Is the resistance of the payload data against either **non**-intentional manipulation of the cover data (e.g. format conversion, compression, A/D conversion – printing + scanning or loadspeaker + recording) or **non**-embedding specific attempts to destroy or remove the payload (e.g. noise, filtering, cutting, attack tools like STirMark or Checkmark). For image integrity verification, WM must not be robust, they need to be *(semi-)fragile*.
- Perceptability: In most cases, a human observer must not be able to perceive embedded data, i.e. he/she should not be able to tell if this is cover data with or without embedded data. For some applications, perceptability can be a desrired feature, e.g. in pre-view images with low quality or to prevent usage of displayed examples (e.g. Vaticane library example).
- Reversability: Is it possible to reconstruct the cover data after WM information has been extracted (in case the original is no longer available) ? Important e.g. in medical imaging !

Information Hiding & Watermarking: Requirements / Properties II

- Detectability: It can be proven by statistical means that data have been embedded (perceptability does not necessarily imply detectability, e.g. in case of colour manipulation).
- Capacity: Amount of data that can be embedded using a certain technique. High capacity combined with high robustness is often desirable, but impossible to combine with imperceptability in most cases.
- Security: Even in case of known embedding procedure (Kerckhoffs principle) an embedded mark must not be removed or destroyed. This is usually achieved by a key-dependent embedding procedure.. Only the WM owner is able to extract WM information. Further, security means resistance against specific attacks, e.g. coalition attacks (i.e. several images which have been WM are used to remove the mark) or non-invertability (i.e. in case of double- or multiple embeddings the embedding order has to be detectable).

Watermarking: Capacity

In most cases high capacity is desirable. Since this usually implies suboptimal properties (e.g. perceptability, poor robustness), alternative methods minimise capacity for enhancing other properties.

- Zero-bit WM: Is usually employed to achieve maximal robustness. Scenarios in which this approach can be applied inlcude
  - the application in in copy protection systems to differentiate among commerical content (encrypted and WM), free content (plaintext and no mark) and illegitimate content (plaintext and WM, i.e. cracked commerical content). Here it is sufficient to distinguish WM from non-WM content since encryapted and plaintext data can be distinguished as well.
  - the application in copyright protection, where it turns out to be irrelevant to embed the name of the copyright holder into the content. It needs to be proven that the “artwork” has been registered in an official registry by the author, which is a binary information only.
- Multi-bit WM: The classical approach to embed user IDs, data about copyright owner, metadata, etc.

Key Management

1. Symmetric Watermarking: The same key is used for embedding and extraction of the watermark.
2. Assymetric Watermarking: One key k_1 is used for embedding the data, a second key k_2 is used for extraction of the WM. Without access to k_2, it is impossible to know and remove the watermark.
3. Public Watermarking: anyone can detect and extract the watermark, even without knowing the key (there is no key) and the content of the mark (blind scheme !). In Private Watermarking, the watermark key and content needs to be known for extraction.
4. Zero-Knowledge Watermarking: During detection and extraction, the WM key is not transmitted to the detector, so it cannot be compromised.
    - Option 1: Communication between content owner and detector owner prove the presence of the WM.
    - Option 2: Detection in the encrypted domain uisng an encrypted key on encrypted content.

Watermarking: Requirements / Properties III

- Oblivious / Blind WM: In case the original cover medium is not required for WM extraction, the scheme is termed semi-blind or “oblivious”, otherwise the scheme is a non-blind one. Of course, if the original cover is not available, extraction is more difficult – caused by transformation to the cover, extraction may get difficult without the original, therfore, less data can be embedded in oblivious techniques. In case the WM in not available additionally, we talk about “blind” techniques.
- Complexity: Computational cost for embedding and/or extraction. Depending on the application, symmetry can be of importance. Also, implementability in hardware can be of importance.
- Embedding domain: Can embedding be done into a bitstream (e.g. H.264) without decoding or do we need to have access to the (uncompressed) raw data. Also the question of embedding in the spatial vs. a transform domain can be an issue.

Watermarking: Application scenarios for (semi-)Blind Schemes

- **Copy-contol**: In case we want to extract copy-control information as embedded vi watermarking, it makes no sense to require the unmarked cover to be present in the device, bcause in this case the customer would already have access to a version without copy-control information.
- **Annotation WM**: If medium with embedded annotation information would be sold, non-blind extraction would require the unmarked original to be available as well causing high transmission overhead.
- **Biometrics**: In case sensor data are subject to WM (to confirm authenticity of the data and to prevent misuse by a replay attack) before transmitting them, the transmission of unmarked data to enable WM detection and extraction is a security risk of course.

Robust Watermarking: Embedding Techniques

- Amplitude Modulation: WM information is embedded by changing pixel values in the spatial / pixel domain. Robustness is low.
- Patchwork Embedding: Data are partitioned into blocks, the properties of which are purposefully changed the information to be embedded by the WM process. Capacity is low.
- Spread Spectrum Embedding: In a transform domain, WM information is embedded into important coefficients (a signal with narrow spectrum, the WM, is spread across several frequencies by embedding it into several coefficients). Important coefficients are selected since these are conserved during imae manipulations. High robustness but low capacity.
- Quantisation-based Embedding: WM information is embedded by empoying different quantisation strategies (WM information steers the choice of the quantiser), which can be identified during WM extraction. High capacity but low robustness.

WM information is usually encoded into Gaussian random numbers (where the WM information is e.g. the seed of the generator) or bi-polar WM sequences (0/1 or -1/1 sequences).

Robust Watermarking: Amplitude Modulation

The most obvious idea is to embed data into the LSB data of perceptual irrelevant colour channels, since in this case perceptability is no problem. However, these techniques are more seen in stego since these are not robust – even slight compression destroys the LSB information.

Algorithm of Kutter

A WM bitstring is embedded into the blue channels by adding a fraction of the luminance channel. Embedding is done in redundant manner, i.e. the string is embedded multiple times:

$$b'(x,y) = b(x,y) + \alpha l(x,y) \text{ if } s_i = 0$$

$$b'(x,y) = b(x,y) - \alpha l(x,y) \text{ if } s_i = 1$$

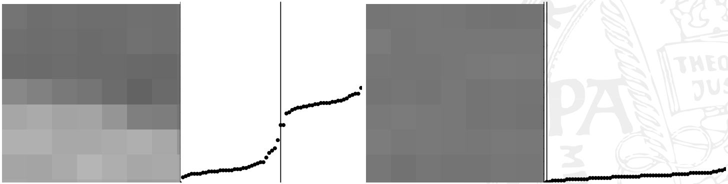
α controls the embedding strength. This approach can be used in an oblivious setting: A star-shaped neighbourhood of a WM pixel is considered during WM extraction which is used for computing a prediction for the central pixel. Depending on the difference to the prediction, corresponding WM bit is extracted. Due to redundant embedding, for each WM bit several predictions are available which facilitates semi-blind extraction of the WM information.

Robust Watermarking: Patchwork Techniques I

As an example, we consider the algorithm of Bruyndoncks: A WM bitstring is embedded into image blocks in the following manner:

- Classification of the blocks' pixels into two zones with more or less homogeneous luminance.
- Subpartition of each zone into two categories which are given as a (key-dependent) patchwork.
- Manipulation of the luminance values for each category in each zone.

For the partitioning of the blocks, two types of blocks are distinguished: (a) blocks with hard or progressive contrast and (b) blocks with noisy contrast.



Robust Watermarking: Patchwork Techniques II

Zones do not need to be contigeous and do not need to contain an equal number of pixels. For the classification, luminance values are sorted and the point with maximal slope in the plots can be chosen to split pixels inti two zones. Blocks with hard contrast are treated this way, while noisy cantrast blocks are partitioned into two equally sized zones.

The patchwork in the figure below is used for creating the two categories – patchwork is secret, key-dependent and can eventually be changed from block to block.

A	A	B	B	A	A	B	B
A	A	B	B	A	A	B	B
B	B	A	A	B	B	A	A
B	B	A	A	B	B	A	A
A	A	B	B	A	A	B	B
A	A	B	B	A	A	B	B
B	B	A	A	B	B	A	A
B	B	A	A	B	B	A	A

B	B	B	B	A	A	A	A
B	B	B	B	A	A	A	A
B	B	B	B	A	A	A	A
B	B	B	B	A	A	A	A
A	A	A	A	B	B	B	B
A	A	A	A	B	B	B	B
A	A	A	A	B	B	B	B
A	A	A	A	B	B	B	B

Robust Watermarking: Patchwork Techniques III

The pixels in the blocks are now subdivided into 5 sets: Those which have been ommitted (due to strong progressive contrast), the remaining pixels are in zone one or two in category A or B. The pixels in each set are counted ($n_{1A}, n_{1B}, n_{2A}, n_{2B}$) and their mean luminance is computed: $l_{1A}, l_{1B}, l_{2A}, l_{2B}$.
Due to construction it is obvious that

$$l_{1A} < l_{2A} \text{ and } l_{1B} < l_{2B}$$

Now, WM bits are embedded as follows:

$$\left.\begin{array}{l} l'_{1A} - l'_{1B} = +\alpha \\ l'_{2A} - l'_{2B} = +\alpha \end{array}\right\} \text{if} \quad s = 1$$

$$\left.\begin{array}{l} l'_{1A} - l'_{1B} = -\alpha \\ l'_{2A} - l'_{2B} = -\alpha \end{array}\right\} \text{if} \quad s = 0$$

Robust Watermarking: Patchwork Techniques IV

In order to conserve perceptual properties, the luminance mean should be kept constant:

$$\frac{n_{1A} l'_{1A} + n_{1B} l'_{1B}}{n_{1A} + n_{1B}} = l_1 \text{ and } \frac{n_{2A} l'_{2A} + n_{2B} l'_{2B}}{n_{2A} + n_{2B}} = l_2$$

To achieve this, luminance of each pixel in a zone is changed by the same value, e.g. pixels in zone 1, category A are changed by $|l'_{1A} - l_{1A}|$.

Robust Watermarking: Patchwork Techniques V

For extracting a WM bit, the same algorithm is used, however, instead of manipulating the mean values these are computed and the differences are determined:

$$s''_k = \begin{cases} 0 & \text{if } l''_{1A} - l''_{1B} < 0 \text{ und } l''_{2A} - l''_{2B} < 0 \\ 1 & \text{if } l''_{1A} - l''_{1B} > 0 \text{ und } l''_{2A} - l''_{2B} > 0 \end{cases} \qquad (4)$$

Now we can apply standard technique to compare the reconstructed WM S'' with the origianal S. Robustness of this algorithm is still limited, but key-dependency is implemented. If the key is known, the WM does not need to be known to be extracted from the data (public WM).

Robust Watermarking: Spread Spectrum Methods (Transform domain) I

Typically, the WM consists of a Gaussian distibuted sequence of n numbers s_i with mean 0 and variance 1. To embed an element of this WM sequence into a coefficient c_i, we proceed as follows (where c_i' is the coefficient carrying the WM information and α is the embedding strength):

$$c_i' = c_i(1 + \alpha s_i) \tag{5}$$

For extracting the WM information, we compute (where c_i^* is the extracted coefficient and s_i^* is the extracted WM information):

$$s_i^* = \frac{c_i^* - c_i}{\alpha c_i} \tag{6}$$

Subsequently, we compute normalised correlation between the extracted WM sequence s^* with the original one ($\overline{s_i}$ is the man of s_i) :

$$C(s, s^*) = \frac{\sum(s_i - \overline{s_i})(s_i^* - \overline{s_i^*})}{\sqrt{\sum(s_i - \overline{s_i})^2}\sqrt{\sum(s_i^* - \overline{s_i^*})^2}}$$

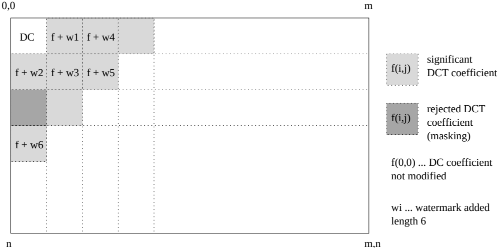
The Algorithm of Cox

This is THE classical algorithm to embedd copyright-protecting WM information (developed at NEC Research & MIT): after choosing the 1000 largest DCT coefficients $f(m,n)$ a Gaussian distributed WM sequence of equal length w_i is added:

$$f'(m,n) = f(m,n)(1 + \alpha w_i)$$

As before, we require the original image to be present (which is DCT transformed and coefficient-wise the WM is extracted). The idea is to embed WM bits into perceptually “important” image regions (described by large coefficients) which are not significantly modified by compression and similar modifications.
As long as image information is conserved to a large extent (= economic value of the image is maintained), the WM should be extractable, since the coefficients are not significantly changed.

The Algorithm of Cox II



0,0
m
DC
f + w1
f + w4
f + w2
f + w3
f + w5
f + w6
n
m,n
f(i,j)
significant DCT coefficient
f(i,j)
rejected DCT coefficient (masking)
f(0,0) ... DC coefficient not modified
wi ... watermark added length 6

This algorithm is highly robust. In case it should be performed as oblivious procedure, the WM detection problem is formulated as signal detection is noise (where the cover data interpreted as noise and the signal is the WM). In this case, interference between cover data and signal is a problem. Thus, for a successful application, the statistical distribution of the cover data needs to be modeled as precisely as possible, to be able to construct reliable detectors (e.g. generalised Gaussian, Laplace distribution for coefficients).

Only for Gaussian distributed data, correlation is the optimal detector (e.g. DC coefficients ar wavelet LL subband).

Wavelet Spread Spectrum Algorithms I

This algorithm transfers the Cox approach to the wavelet domain. The DWT is iterated, then the approximation subband coefficients are manipulated (where $\overline{c_l}$ is the mean of the approximation coefficients $c_{l,i}$).

$$c'_{l,i} = \overline{c_l} + (c_{l,i} - \overline{c_l})(1 + \alpha s_i)$$

with s_i being a WM bit.
For extraction, the same transform is conducted, mean $\overline{c}^*_l$ and variance $\sigma(c^*_l)$ of the coefficients is computed, and the reconstructed WM bit s^*_i is determined as

$$s^*_i = \frac{(c^*_{l,i} - \overline{c^*_l})\frac{\sigma(c_l)}{\sigma(c^*_l)} - (c_{l,i} - \overline{c_l})}{c_{l,i} - \overline{c_l}}$$

The consideration of mean and variance make the algorithm robust against changes in contrast and luminance.

Wavelet Spread Spectrum Algorithms II

The algorithm of Wang is based on embedding WM information into the n most significant detail coefficients. The selection of these coefficients is done iteratively. For each subband k at decomposition level l the larges coefficient-value $\hat{c}_{k,l}$ is determined. Based on this, a treshold $T_{k,l}$ is chosen for each subband:

$$T_{k,l} = \frac{1}{2}\beta_{k,l}\hat{c}_{k,l}$$

with $\beta_{k,l} \in (0, 1]$ parameters dependent on each subband. For initialisation, a set of candidate coefficients U is selected containing all detail coefficients. Subsequently, the following iteration is conducted until a sufficiently large number of coefficients is selected:

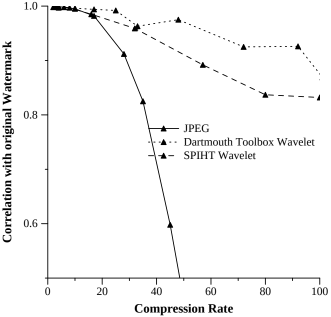
Wavelet Spread Spectrum Algorithms II

1. Select the subband with maximal value for $T_{k,l}$.
2. Coefficients in the selected subband still contained in U and being larger than $T_{k,l}$ are removed from U and are subject to WM embedding.
3. $T_{k,l}$ is divided by 2.
4. Continue with iteration until all WM bits are embedded.

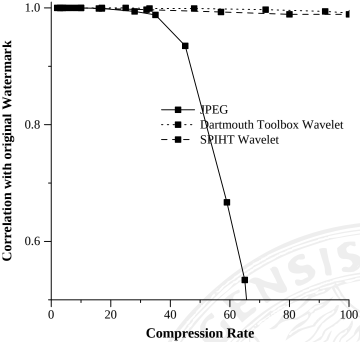
For the actual embedding, the following formula is used:

$$c'_{k,l,i} = c_{k,l,i} + \alpha\beta_{k,l} T_{k,l} s_i$$

Spread Spectrum Embedding: Compression Robustness



Watermark cox(0.0475) under compression



Watermark wang(0.275) under compression

Robustness against compression depends on the technique used, but can be fairly high. It is discussed somehow controversial if it is beneficial to employ the same strategy (e.g. transformation, notion of significance) for compression and WM or not.

Robust WM: Quantisation Embedding I

In these algorithms, the signal range is separated into distinct partitions, which are mapped onto the set of values to be embedded by a function $Q()$ (e.g. to $b \in \{0, 1\}$ for binary WM informations).

Let I_0 are the original data and I_1 are the marked data, we select I_1 from the partition of the signal which is mapped to b such that $b = Q(I_1)$ holds, which is equivaluent to the extraction process.

Additionally, I_1 is selected "as close as possible" to I_0 to limit the impact on perception.

This is a fundamental difference to the spread spectrum approach: WM information b is embedded by addition or multiplication into the data; in order to embed a bit, the difference between I_0 and I_1 is a function of b, i.e. $I_1 - I_0 = f(b)$. Thus, contrasting to the idea above, I_0 is important and knowing I_0 improves WM extraction.

Robust WM: Quantisation Embedding II

There are several options how to separate the signal into appropriate partitions. One possibility is to consider certain n-tupel of coefficients. The map $Q()$ can be applied to magnitude-relations among these n-tupel.
A classical example is the Algorithm of Koch [24]: Following a block-based DCT, two coefficients v_1, v_2 are selected from each block (following perceptual criteria). These coefficients are (eventually) manipulated into coefficients v_1', v_2', such that $v_1' > v_2'$ for $b = 1$ and $v_1' < v_2'$ for $b = 0$. It is essential, that v_1, v_2 and v_1', v_2' are perceptually close. Similarly, we might select triples of coefficients or coefficient signs.
In order to achieve robustness, embedding has to enforce a tolerance band with size A, such that $|v_1' - v_2'| \geq A$ (for order relations between two values) or $|v'| \geq A$ (sign of a value).
A further simple example is "Even-Odd Embedding", where the even number being closest to the sample is used to embed a "0" and the closest uneven number is used to embed a "1". Extraction is a simple check for even/odd, the band of tolerance is 1 symmetric around each number. THis procedure is similar to LSB embedding by he way and not very robust.

Quantisation Index Modulation (QIM) I

QIM generalises Even-Odd embedding by defining several quantisers. During embedding, the WM value to be embedded decides which quantiser is chosen for a given data value. During exraction, the closest quantiser reconstruction point is searched: The corresponding quantiser determines the extracted WM information.
In the following we consider a scalar quantiser as a simple example (in the actual example using quantiser-stepsize $q = 2$):

$$Q(x) = q\lfloor\frac{x}{q}\rfloor \text{ or in the symmetric case} Q(x) = q\lfloor\frac{x+q/2}{q}\rfloor$$

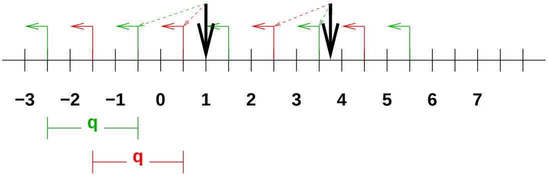
We use the simpler non-symmetric case and define two quantisers ($+-q/4$ is termed dither vector) for QIM as follows:

$$Q_0(x) = q\lfloor\frac{x+q/4}{q}\rfloor - q/4 \text{ and} Q_1(x) = q\lfloor\frac{x-q/4}{q}\rfloor + q/4$$

QIM II

Setting $q = 2$ we get:

$$Q_0(x) = 2\lfloor \frac{x + 1/2}{2} \rfloor - 1/2 \;\; \text{bzw.} \;\; Q_1(x) = 2\lfloor \frac{x - 1/2}{2} \rfloor + 1/2$$

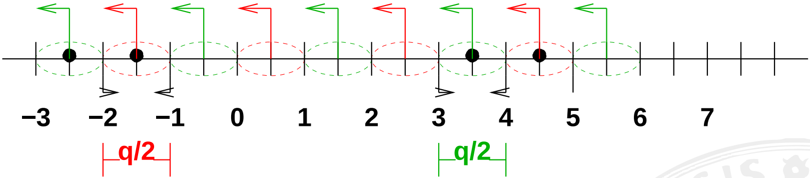


Let us consider first the green quantiser Q_0: $Q_0(0) = -1/2$, $Q_0(1) = -1/2$, $Q_0(2) = 11/2$, $Q_0(3) = 11/2$, $Q_0(4) = 31/2$, $Q_0(5) = 31/2$, and so on, which means that in the green intervals all values are quantised into the left corner value of the green interval. Similarly we get for Q_1 (the red quantiser): $Q_1(0) = -11/2$, $Q_1(1) = 1/2$, $Q_1(2) = 1/2$, $Q_1(3) = 21/2$, $Q_1(4) = 21/2$, $Q_1(5) = 41/2$, and so on, which means that also in the red intervals all values are quantised to the left corner point of the interval.

Depending on the WM bit to be embedded, Q_0 or Q_1 is applied to a data value (large black arrows).

QIM III

WM extraction is done by choosing the quantiser reconstruction point, which is closest to the extracted data value. The quantiser being “responsible” for that data value determins the extracted bit.



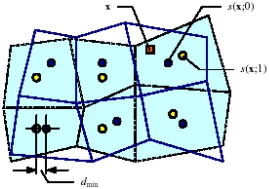
In case of no modification of I_1. the WM can be extracted without inaccuracy. In case of modification, each point of reconstruction is surrounded by a tolerance band sized $d/2$. In this context, the tradeoff between robustness and perception is seen very drastically: the larger the quantiser step size, the larger the tolerance band, but also the stronger quantisation gets.

QIM IV

The general idea can be applied to arbitrary quantisers $Q_m(s)$, $m \in \{0,1\}$, where each Q_m is a map from the real-values axis into a codebook $B_m = \{b_{1,m}, b_{2,m}, b_{3,m}, b_{4,m}, \cdots, b_{L,m}\}$. The results of the quantisers $b_{i,m}$ $1 \leq i \leq L$ are deonoted as reconstruction points. When the receiver obtains the sample y of signal I_1, the WM bit is obtained by:

$$m(y) = argmin_{m \in \{0,1\}} |y - b_m|$$

This idea can also be applied to vector quantisation, in various transform domains, and in various other variants.



Robust Watermarking Security: Application Scenarios

1. An advertizing company embeds WMs into its clips broadcasted. The broadcasted content is recorded and the WMs are extracted automatically in order to verify the amount of money charged by the broadcasting company for sending the clips. The broadcasting company may embed the WM of the advertizing company into any arbitrary material broadcasted and may charge the advertizing company for that. This is **Unauthorized Embedding** or **forgery attack** or **copy attack**.
2. A WM service provider A embedds WMs into images and offers search for corresponding material on the web. A second WM service proider B only offers a (cheaper) search for marks embedded by A. This is **Unauthorized Detection** or **passive attack**.
3. A movie production company embeds WMs into the movies before distribution to prevent compliant recorders from copying. A movie priat removes these WMs and is subsequently able to copy the movies, even with compliant rcorders. This is **Unauthorized Removal**.

Robust WM Security: Unauthorized Embedding I

- Copy Attack: Identical WM is transfered to other (unmarked) data.
- In the more general case, a new message is encoded as WM and is embedded into cover data.

In case of encrypting the WM using a secret key, the second attack type can be avoided, but not the copy attack, since an encrypted WM can be copied in the same manner as an unencrypted one. In order to prevent this type of attack, a relation between WM and cover data needs to be established, e.g. the WM could contain a signature of the cover image.

Problem: Caused by Embedding, the cover image is modified, such that the signature embedded would no longer be valid.For this application, robust image features are required, which are invariant w.r.t. WM embedding (e.g. low-frequency DCT coefficients).

Robust WM Security: Unauthorized Embedding II

The procedure sketched before is conducted as follows:

1. Generate a description of the cover data based on robust features (see robust media hashes)
2. The WM is concatenated with the robust features and a classical hash is computed from these data.
3. This hash is encrypted using a private key and represents a signature.
4. The WM is embedded together with the signature. Embedding must not change the robust features.

Robust WM Security: Unauthorized Detection & Unauthorized Removal

Unauthorized Detection can be limited to recognition of embedded data (which is the pure steganographic application case) or may include decoding of WM information as well (which can be prevented by applying encryption to WM content). As an in-bewteen scenario, an attacker may want to distinguish among different WMs.
Unauthorized Removal is successful, in case the WM cannot be extracted any more, but the cover data is of high quality. We may distinguish between *elimination attack* and *masking attacks* (e.g. by applying synchronisation attacks like slight rotation). In [11] we were able to prevent both types of attacks by applying key-dependent embedding domains (i.e. parameterised wavelet filters as used for encryption).

Watermark Security: Further attacks I

- Scrambling Attack: In case parts of the cover data are encrypted, a WM detector is disabled. Before the image data is used, scrambling has to be inverted. For example, a permutation of 8x8 pixels blocks will not be sufficient to conceal WM which are embedded independently into 8x8 blocks. Can be used to circumvent copy control mechanisms: The input into the recording device is encrypted such that copy control WMs are concealed. During decoding, the signal is decrypted before being transmitted to the viewer.
- Mosaic Attack: Specific attack for which images are tiled into blocks which are too small for detecting a WM. For display, these tiles are presneted as components of a table such that no difference to the original is visible.

Watermark Security: Further attacks II

False positive detection is performed in case a WM is detected but no WM has been embedded. In case an image already contains a WM before embedding an other one this is termed “WM collision”. Both effects lead to problems in case of content ownership WM and questions reliability of such systems.

The *Ambiguity attack* [8] relies on two WMs that can be embedded into one image. Alice, the legitimate image owner, generates the watermarked image $\tilde{I} = I + w$ from her original image I and the WM w. $\tilde{I}$ is passed on to clients. In case an image in question I' is investigated for copyright information, Alice computes $I' - I = x$ and correlates x and the originally embedded w: $c(w, x)$.

Mallory now also claims to be the legitimate owner of the image. He computes $I' = \tilde{I} - x = I + w - x$ and claims I' to be the original. Alice is able to prove her WM in Mallory’s image: $I' - I = w - x$, $c(w, w - x) = 1$ for any random pattern x. Since two WMs can be embedded, the substraction hardly interferes with the correlation. On the other hand, Mallory is able to prove his WM in the original of Alice: $I - I' = x - w$, $c(x - w, x) = 1$, as well as in the copies distributed by Alice: $\tilde{I} - I' = I + w - (I + w - x) = x$, $c(x, x) = 1$. The attack relies on the invertability of the WM scheme. This can be prevented by designing WM embedding to be a one-way function of the original image.

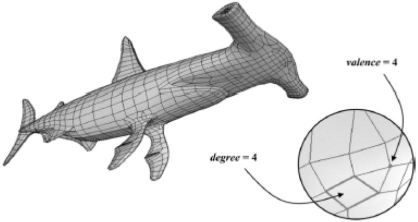
Watermark Security: Further attacks III

- Oracle/Sensitivity Attack: Relies on the existence of a public WM detector with binary output. The attacks starts by constructing an image close to the threshold of the detector. Subsequently, the sensitivity of the WM scheme towards modifications of single pixels is probed, resulting in the ability to determine the modifications with least impact on perception and maximal impact on detectability.
- Gradient Attack: Relies on the existence of a public WM detector with detection value output. Gradient decent techniques are used to determine optimal modifications (as before).

These attacks illustrate the problems when public WM detectors are available.

Robust WM for 3D Graphics Data: 3D-meshes I

By analogy to partial encryption for 3D graphics data, we now also consider the “the second line of defence” in DRM, i.e. robust WM for 3D meshes (since 3D meshes are the most general representation of graphics data in which all other representation forms can be converted, almost no other literature is available). There is much more literature available as compared to encryption of 3D data, but on the other hand, much as compared to audio, image, or video WM.



The valence of a vertex is the notion for the number of edges ending in a vertex (or equivalent the number of direct neighbours), the degree of a face is the notion for the number of edges “surrounding” the face.

Robust WM for 3D Graphics Data: 3D-meshes II

Reasons for the lower number of techniques as compared to classical WM media types are on the one hand difficulties caused by the topology with more degrees of freedom and the irregular sampling of meshes, and on the other hand the numerous ways to attack such schemes. WM-relevant differences to classical media include:

- There is no obvious intrinsic ordering of data (e.g. vertices and faces), these can be re-ordered in many ways. In classical media, order of WM-carrier is used to synchronise embedding and detection.
- Caused by irregular sampling, spectral analysis using e.g. DCT, Fourier, DWT ect. is much more complicated. In classical media, these transforms are essential WM tools.
- A 3D object can be represented in various ways in terms of geometry and connectivity informations, e.g. a mesh with 1-to-4 connectivity (each vertex has 4 neighbours) or with 1-to-6 connectivity. A robust WM should be present in all ways to represent the data.

Attacks against Geometry in 3D-Mesh WM

- Similarity transforms: Translation, rotation, scaling and combinations of these techniques. Countermeasures are WM-primitives (i.e. data in which the WM is embedded) which are invariant against these operations (e.g. relation among height and edge length in triangles), synchronisation techniques being insensitive against these operations (e.g. a specified scan ordering w.r.t. the length of the edges of a triangle), WM being applied in a space invariant against these operations and registration w.r.t. the original data using non-oblivious WM.
- Signal Processing attacks: Random noise addition, smoothing, compression / quantisation. Similarly to the first point these operations are conducted in daily routine in animation and special effects and eventually remove the WM. Since all these operations take place in the "higher frequencies", a natural countermeasure is to apply WM in low and medium frequencies. Also statistical mesh-features can be of help.
- Local Deformation: Can lead to severe synchronisation problems - a segmentation into local sub-meshes with redundant WM embedding may serve as countermeasure. Segementation can hardly be done in oblivious manner.

Attacks against Connectivity Information in 3D-Mesh WM

- Cropping: Quite similar to local deformation in fact, thus, a countermeasure is again redundant embedding into sub-meshes.
- Remeshing and Simplification: Only WM techniques can be robust, which employ WM primitives representing the overall shape of the mesh. Histogramm-based techniques can be of help, in case of non-uniform remeshing or subdivisions these fail as well however. A solution is found in mesh restauration, which tries to reconstruct the identical original connectivity by remeshing.
- File Attack (reordering of vertices or facets in the describing file) and representation conversion (i.e. approximation of the data by a NURBS model or voxel such that the mesh no longer exists) cannot be handeled with current techniques.

3D-Mesh Transformations: Spectral Analysis I

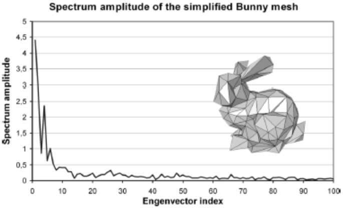
Since it does not make sense to apply a transformation directly to vertex or connectivity information, the mesh Laplace matrix is used instead, which contains both geometry and connectivity informations. In many cases, the "combinatorial Laplacian" of "Kirchhoff Matrix" K is formed: $K = D - A$, where D is a diagonal matrix, holding the valence of vertex i (the number of direct neighbours) as its diagonal elements and A is a neighbourhood matrix with elements $a_{ij} = 1$ in case vertices i and j are direct neighbours and $a_{ij} = 0$ otherwise. Thus, a mesh with n vertices results in a matrix K with $n \times n$ elements. Subsequently, a eigenvalue decomposition is conducted, $K = Q^t \;\; \Omega \;\; Q$, where Ω is a diagonal matrix with n eigenvalues as diagonal elements and Q is the matrix of eigenvectors w_i. The pairs of eigenvalues / eigenvectors are sorted according to the size of the eigenvalues (can be seen as corresponding to increasing frequency).

After sorting, each component of the Cartesian vertex coordinates $v_i = (x_i, y_i, z_i)$, $1 \leq i \leq n$ is projected separately onto the ith normalised eigenvector $e_i = \frac{w_i}{|w_i|}$ which generates n vectors of mesh spectral coefficients $r_i = (r_{i,x}, r_{i,y}, r_{i,z})$.

3D-Mesh Transformations: Spectral Analysis II

Thus, by analogy to classical frequency analysis we obtain:

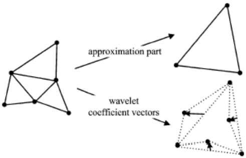
$$
\begin{aligned}
(x_1, x_2, \ldots, x_n)^T &= r_{1,x} e_1 + r_{2,x} e_2 + \cdots + r_{n,x} e_n \\
(y_1, y_2, \ldots, y_n)^T &= r_{1,y} e_1 + r_{2,y} e_2 + \cdots + r_{n,y} e_n \\
(z_1, z_2, \ldots, z_n)^T &= r_{1,z} e_1 + r_{2,z} e_2 + \cdots + r_{n,z} e_n
\end{aligned}
$$

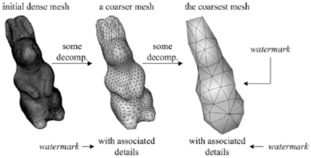


The representation of the spectral domain (large amplitude for lower frequencies) is also similar to a classical power spectrum. The problems with this representation are the high computational demand (eigen value decomposition is expensive due to the size of the matrix, here efficient algorithms and submeshing helps) and the dependence on the connectivity information (here remeshing by analogy to the original helps).

3D-Mesh Transformations: Wavelets

Wavelet analysis of 3D meshes is conducted using the “lazy wavelet decomposition”: In the example 4 triangles are fused into a single one, only 3 out of 6 original vertices are retained in the lower resolution. Wavelet coefficients are computed as prediction errors of the lost vertices, i.e. these coefficients are 3D vectors associated with each edge of the mesh with lower resolution. A certain mesh regularity is required for this procedure.





By analogy to the spectral decomposition, WM can be embedded into different frequencies (which are actual resolutions in this case), with different implications on robustness. Similar techniques have been developed for progressive meshes.

Spectral transform-based WM for 3D Meshes I

In most proposals the lower frequency half is proposed for embedding, since these represent a good approximation of the overall mesh shape. By repetition, a sufficiently long WM sequence of WM-bits $b_i \in \{-1, 1\}$ is generated, additionally a WM-key $p_i \in \{-1, 1\}$ is used (e.g. produced by a PRNG). WM embedding is done in each coordinate of the frequency vector $r_i = (r_{i,x}, r_{i,y}, r_{i,z})$ by:

$$\hat{r}_{i,x} = r_{i,x} + b_i p_i \alpha$$

where α is the embedding strength. Subsequently, the WMed mesh is reconstructed:

$$(\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n)^T = \hat{r}_{1,x} e_1 + \hat{r}_{2,x} e_2 + \cdots + r_{n,x} e_n$$

$$(\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n)^T = \hat{r}_{1,y} e_1 + \hat{r}_{2,y} e_2 + \cdots + r_{n,y} e_n$$

$$(\hat{z}_1, \hat{z}_2, \dots, \hat{z}_n)^T = \hat{r}_{1,z} e_1 + \hat{r}_{2,z} e_2 + \cdots + r_{n,z} e_n$$

Spectral transform-based WM for 3D Meshes II

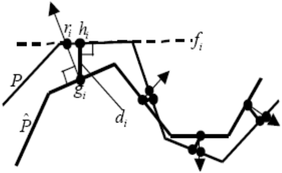
WM extraction is done as in classical spread spectrum WM (using the original mesh). Both meshes undergo the spectral transform, and the reference coefficients $r_i = (r_{i,x}, r_{i,y}, r_{i,z})$ as well es the eventually WMed coefficients $\hat{r}_i = (\hat{r}_{i,x}, \hat{r}_{i,y}, \hat{r}_{i,z})$ are computed.

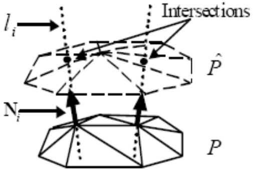
Subsequently, the difference $(r_i - \hat{r}_i)$ is multiplied by the embedding key $p_i \in \{-1, 1\}$. The result is correlated with the original WM sequence, where additional redundancy is obtained by embedding into all three coordinate axes.

Since the computation of the spectral coefficients directly relies on the Cartesian coordinates of the meshes, a correct alignment or remeshing is required, to facilitate a synchronised readout of the WM bits:

Spectral transform-based WM for 3D Meshes: Alignement/Re-meshing

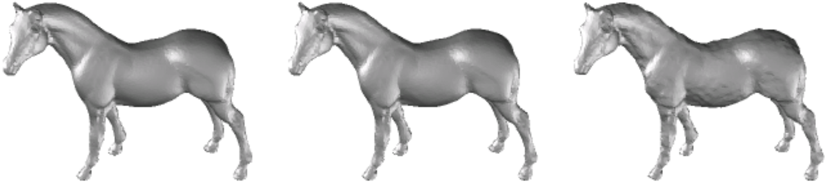
After a coarse alignment (either manually or automated using the axes of point clouds distributed randomly across the mesh) a fine adjustment is done by an iterative minimisation of the sum of distances among original and WMed mesh (P and $\hat{P}$). For this purpose, the expression $\sum_i \sqrt{(}g_i - h_i)^2$ is minimised, where g_i is the center point of a triangle edge on $\hat{P}$, r_i is the intersection of an orthogonal line to g_i with a triangle edge of P, and h_i is the intersection of another orthogonal line to the triangle edge of P with $\hat{P}$.





After alignment a resampling is done: From each vertex on P, a line in the direction of the normal vector is intersected with $\hat{P}$. The normal vector is computed as the mean of all triangle normal vectors adjacent to the considered vertex. In case of no close intersection, the vertex of P is used.

Spectral transform-based WM for 3D Meshes: Example



(a) Original model

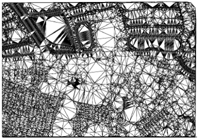
(b) Watermarked with $\beta = 0.001$.

(c) Watermarked with $\beta = 0.005$

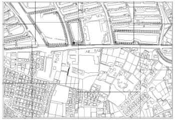
In the figure, increasing WM strength is visualised. Artifacts look like a rough, knobby surface.

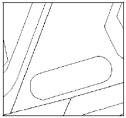
Spectral transform-based WM for 3D Meshes: Application

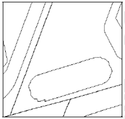
A planar application is WM of maps. As a first step, a Delaunay triangulation is computed from the digitised map to generate a single connected mesh.



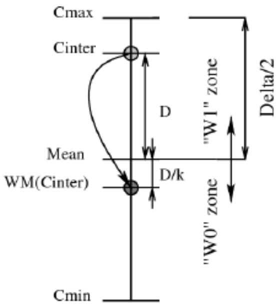
Subsequently, patches are generated (for reducing complexity and redundant embedding) dependent on the number of local vertices. Subsequently the WM is embedded and the original map connectivity is re-generated.







Blind Spectral Transform-based WM for 3D Meshes



We embed a WM bit into each coefficient triple $r_i = (r_{i,x}, r_{i,y}, r_{i,z})$ using quantisation WM, preferably using low and medium frequencies for robustness: After computing $a = \max(r_{i,x}, r_{i,y}, r_{i,z})$ and $b = \min(r_{i,x}, r_{i,y}, r_{i,z})$, we compute $mean = \frac{a+b}{2}$, which is used to identify the embedded WM bit: In case a zero is embedded, the middle coefficient is in $[b, mean]$, otherwise in $[mean, a]$.

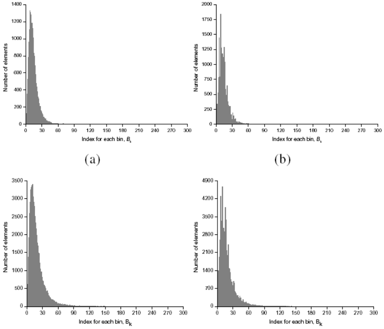
The distance between *mean* and the eventually changed coefficient determines the robustness (which is low as in all blind schemes), on average only every second coefficient needs to be changed / quantised.

Mesh WM using Laplace Coordinate Histograms I

After computing the Kirchhoff matrix K, for each vertex $v_i = (x_i, y_i, z_i)$, $1 \leq i \leq n$ the Laplace coordinates $L_i = (x'_i, y'_i, z'_i)$ are computed by $L = K \times O$; the n lines of O contain the n vertices in their Cartesian coordinates. For each vertex we compute $d_i = \sqrt{(}x'^2_i + y'^2_i + z'^2_i)$, all d_i are mapped to histogram bins. These histograms (or pairs of their bins) serve as WM primitives.

The bipolar WM $\in \{-1, 1\}$ is embedded into a "valid" pair of bins (B_{k1}, B_{k2}) (i.e. neighbouring bins are used containing a sufficient number of elements, avoiding the first and the last bin) by: $|\hat{B}_{k1}| < |\hat{B}_{k2}|$ for embedding a -1, and vice versa for 1. This means that single d_i are changed such that they change their corresponding bin. Note that manipulation is required only in case the relation is not yet fulfilled.

Mesh WM using Laplace Coordinate Histograms II



(a) (b)

(c) (d)

For WM embedding, the required number of smallest elements of B_{k+1} is moved to B_k by $\hat{d}_i = d_{mean}$, where d_{mean} is the mean of the d_i's of B_k. Subsequently, manipulated Laplace coordinates $\hat{L}_i$ are computed with $||\hat{L}_i - L_i||^2 = (\hat{x}'_i - x'_i)^2 + (\hat{y}'_i - y'_i)^2 + (\hat{z}'_i - z'_i)^2$ such that $\hat{x}'^2_i + \hat{y}'^2_i + \hat{z}'^2_i = \hat{d}^2_i$.

It turns out that $\hat{x}'_i = \frac{x'_i \hat{d}_i}{\sqrt{(x'^2_i + y'^2_i + z'^2_i)}}$, and $\hat{y}'_i, \hat{z}'_i$ is computed by analogy. Finally, the Cartesian coordinates of the mesh are computed by $K^{-1} \times \hat{L}$. Extraction can be done blind and follows the embedding strategy.

Mesh WM in Spherical Coordinates I

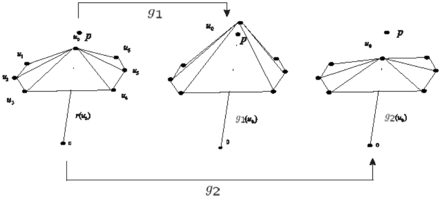
For each $v_i = (x_i, y_i, z_i)$ the sperical coordinates $v_i^s = (r_i, \Phi_i, \phi_i)$ are computed as follows:

- Translation of the mesh center of gravity into the coordinate origin.
- Alignment of the main axis: The 3D mesh is rotated such that the PCA main axis corresponds to the z-axis (the PCA main axis is the eigen vector of the largest eigen value of the co-variance matrix of the vertex coordinates).
- Coordinate conversion (where (x_i, y_i, z_i) are already rotated and translated): $r_i = \sqrt{(}x_i^2 + y_i^2 + z_i^2)$, $\Phi_i = \arccos(z_i/r_i)$, $\phi_i = \arctan(y_i/x_i)$.

Actual embedding only modifies r_i, vertices subject to embedding are selected by a PRNG, which generates an angle $\hat{\Phi}_i$: The vertex v_i^s with Φ_i being closest to the random angle is WMed next.

Mesh WM in Spherical Coordinates II

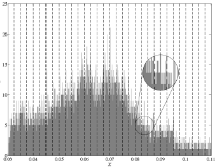
WM embedding is done as shown in the figure. For WM value +1 a function g_1 is applied to the selected vertex P, by setting r_i to a value which is α-times the average or mean value of the r_i values of the direct neighbours, for WM value -1 a corresponding function g_2 is defined.

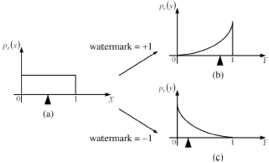


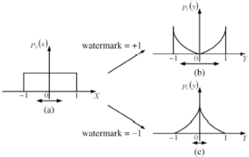
Vertices used for determination of the modification of another vertex are not marked. Exteraction is done in the same manner and compares the actual r_i value with the corresponding value of the neighbours for deriving the WM bit. This technique is not robust against remeshing and simplyfication.

Mesh WM in Spherical Coordinates III

For achieving robustness against remeshing and simplyfication, authors have used histograms of r_i for embedding. To embed n WM bits, n bins are generated and the r_i in each bin are normalised. In each bin (for which uniformy distributed r_i are assumed) a WM bit is embedded by increasing / decreasing the mean (method I) or the variance (method II) in the bin.

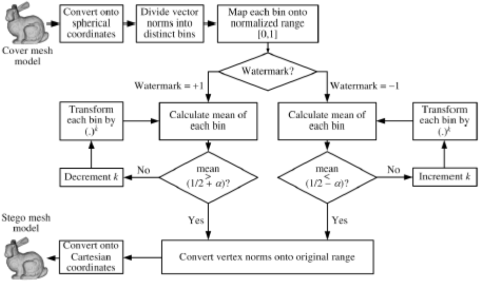


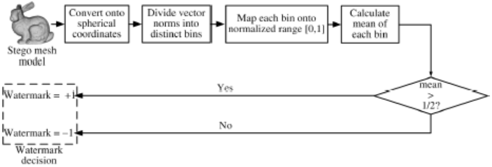




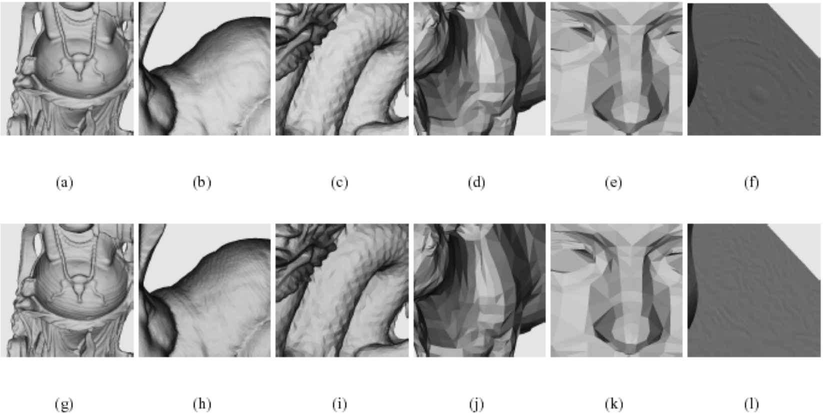
Manipulation of the r_i is done similar to modifying histograms of grayscale images. Subsequently, Carthesian coordinates are reconstructed. WM extraction is done by comparing the expected mean (or variance) of the bin (e.g. mean 1/2 for normalised r_i) to the actual value.

Mesh WM in Spherical Coordinates IV


Cover mesh model
Convert onto spherical coordinates
Divide vector norms into distinct bins
Map each bin onto normalized range [0,1]
Watermark?
Watermark = +1
Watermark = −1
Transform each bin by $(.)^k$
Calculate mean of each bin
Calculate mean of each bin
Transform each bin by $(.)^k$
Decrement k
No
mean $>$ $(1/2 + \alpha)$?
mean $<$ $(1/2 - \alpha)$?
No
Increment k
Stego mesh model
Yes
Yes
Convert onto Cartesian coordinates
Convert vertex norms onto original range


Stego mesh model
Convert onto spherical coordinates
Divide vector norms into distinct bins
Map each bin onto normalized range [0,1]
Calculate mean of each bin
mean $>$ 1/2?
Yes
No
Watermark = +1
Watermark = −1
Watermark decision

Mesh WM in Spherical Coordinates: Examples

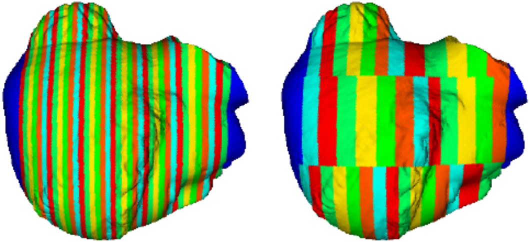


(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

Spectral transform-based WM for 3D Meshes: High Frequency Embedding

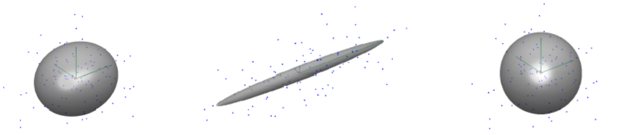
Techniques discussed before embed into mid- and low frequency components due to robustness. As an alternative, it has been suggested to embed into the 70% high frequency coefficients – however, not single coefficients are targeted but the statistical distribution of all high frequency components. After a robust alignment the mesh is partitioned into smaller submeshes, into which a single bit is embedded.



The spectral transform is applied and high frequencies are used.

Spectral transform-based WM for 3D Meshes: High Frequency Embedding II

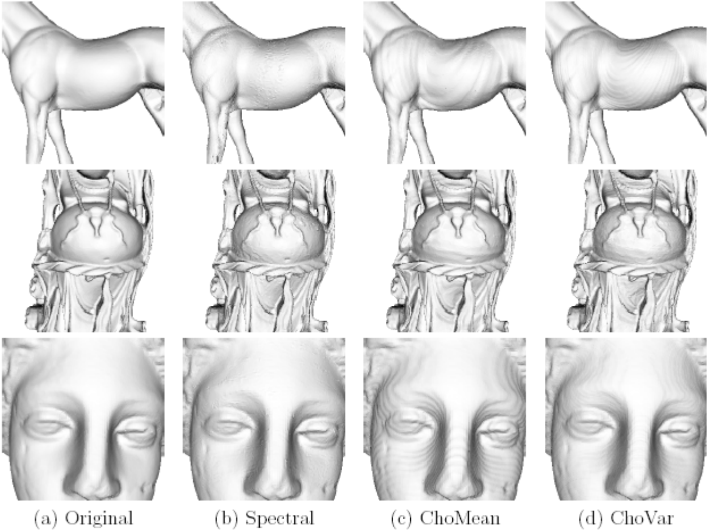
The point cloud fo high frequency coefficients is analysed with PCA,the three eigen values represent the variances in the three axes (which are spanned by the eigen vectors). The point cloud is rotated such that the axes correspond to the three eigen vector directions.



(b) Ellipse of the original coefficients (c) Ellipse with 1 embedded (d) Ellipse with 0 embedded

For embedding, the point cloud is deformed into an ellipse (WM +1) or a circle (WM is 0), which is done by modifying variances of the second and third axes, which leaves the variance w.r.t. the first axis unmodified (i.e. the coefficient vectors are manipulated correspondingly). After backrotation (reconstruction of spectral coefficients) the Carthesian coordinates are reconstructed. Extraction by analogy, but rather sensitiv against alignment and patch generation problems.

Spectral transform-based WM for 3D Meshes: High Frequency Embedding III



(a) Original (b) Spectral (c) ChoMean (d) ChoVar

Steganography: Properties

Similar to WM, data are embedded into a cover medium. Due to the different target appliation, significant differences do exist:

- In WM, usually, the embedded information has something to do with the cover medium
- We have seen several attacks against robust WM systems, in steganography an attacker wants to reveal that data has been embedded. Thus, robustness against cover manipulations is not a necessary requirement (although it might be desirable).
- Capacity is important in stego appliations.

Steganography is often seen as an appropriate answer to restrictions in using strong crypto or against the ECHOLON (and similar) system, especially by citizens' rights groups.

Steganography: History

- Technical steganography
  - Shaving the head, apply a tatoo, regrow the hair (400 v.Chr.)
  - Wax tables: Informations have been hidden behind the wax layer used for writing
  - Invisible ink
  - Informations coded as small holes or dots (e.g. WitnesSoft by Aliroo)
  - Microfilm/Microdots as comma in texts
- Linguistic Steganography
  - Distinct positions in the cover text make up the hidden message (in old China this has been done using masks with cut out positions overlayed to the cover text).

Steganography – Basics

The majority of the available Stego Tools (e.g. StegoDos, SysCop, S-Tools, Mandelsteg, EzStego, Hide and Seek, Hide4PGP, White Noise Storm, Steganos) apply some kind of LSB replacement or manipulation of the colour palette. Thus they are not robust and the embedded data can be removed with tools like *Stirmark*, *Checkmark*, or *UnZign*. Also, these techniques are not even resistant against moderate compression.

An attacker can be *passive* (messages are tested, blocked, no manipulation), *active* (unspecific manipulations since the type of scheme used is unknown) or *malicious* (specific attacks against a particular stego-system, injection of wrong messages). in most cases the passive attacker is assumed, since the other attack scenarios are tackeled in the WM domain.

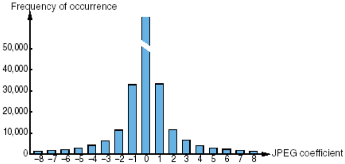
Steganography – Steganalysis

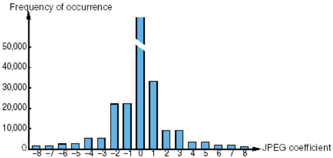
There are three different types of steganalysis, differentiated by varying knowledge of the attacker.

- **Targeted Steganalysis**: The attacker is assumed to have detailed knowledge about the used embedding scheme (i.e. structural weaknesses can be exploited). The aim is not only a binary answer but eventually an estimation of the length of the embedded data.
- **Blind** / **Universal Steganalysis**: Typically, feature extraction with subsequent classification is employed – training data are used to train the classifier to stegoworks and coverworks (2 classes). Since the embedding scheme is unknown, training data of many different stego algorithms are used. In this case is is often better to use a one-class classifier (only coverworks are trained).
- **Forensic Steganalysis**: The aim is to reveal the actual content of the stego message, the embedding scheme or the used key material (stegowork-only attack, known cover attack, known-stego method attack, known message attack).

Steganographic Algorithms: LSB I

LSB embedding is not suited as a robust watermark, but as discussed, this is not an issue for steganography. If we embed a 0 into 51, $(51)_2 = 00110011$, the last bit is replaced by a 0, thus, $00110010 = (50)_2$. Vice versa, 50 is mapped to 51 in case 1 is embedded into the LSB. By analogy, the entire range of values can be structured into “pairs of values” (PoV) $(2i, 2i + 1)$ which are mapped onto each other. Since data to be embedded can be assumed as randomly distributed, PoV are mapped to similar values.





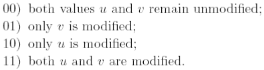
By analogy, LSB embedding can be applied to quantised DCT coefficients (except for the fact that many coefficients are 0 or 1 – which need to be skipped – reducing capacity). One example is JSteg for which the changes in the histogram are clrearly visible (and which can be detected by statistical means, as it is th case for the spatial domain embedding).

Steganographic Algorithms: LSB II

Changes in the histogram caused by LSB embedding can only be detected in case of large payload. An advanced technique to detect LSB embedding and to estimate the message length is denoted “sample pair analysis”. The set P of horizontally adjacent pixels in an image are considered. We define:

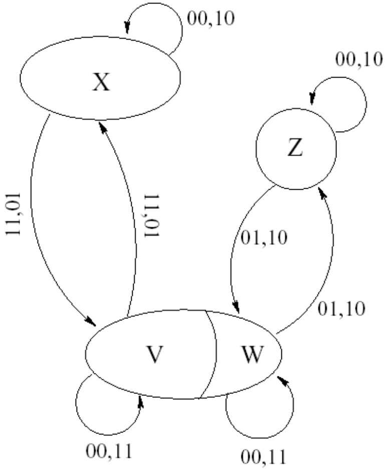
- X is the set of pairs $(u, v) \in P$ such that v is even and $u < v$ or v is odd and $u > v$.
- Y is the set of pairs $(u, v) \in P$ such that v is even and $u > v$ or v is odd and $u < v$.
- Z is the set of pairs $(u, v) \in P$ such that $u = v$.
- Furthermore, Y is devided into to subsets W and V, where W only contains pairs of the form $(2k, 2k + 1)$ and $(2k + 1, 2k)$ and $V = Y - W$. X and Y are assumed to be of similar size (gradients in both directions occur with a similar count).

LSB embedding can cause the following 4 situations (X, V, W, Z are denoted primary sets and there union is P):



00) both values u and v remain unmodified;
01) only v is modified;
10) only u is modified;
11) both u and v are modified.

Steganographic Algorithms: LSB III



Due to embedding, pairs do change their sets. For the different change scenarios the expected number of modified pixel pairs is computed, expressed by *q* wich is the length of the message (payload). For all sets, we express their cardinalities after embedding (e.g. $|X'|$) as functions of their cardinalities before embedding and *q*.

The aim is to express *q* only by relating the set cardinalities after embedding:

$$\frac{\gamma q^2}{2} + (2|X'| - |P|)q + |Y'| - |X'| = 0$$

with $\gamma = |W'| + |Z'|$ and γ is hardly 0 in natural images.

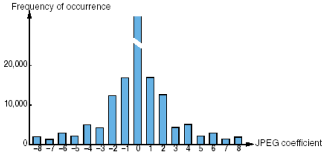
Steganographic Algorithms: Outgess & F3

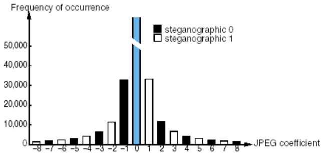
Outguess does LSB embedding on DCT coefficients except for the pair $(0, 1)$ since 0-coefficients cannot be manipulated due to their high number and the visual impact and 1-coefficients would be changed to 0 due to the PoV structure (histogram would lose balance). Not all remaining coefficients are modified but only so many, that the most imbalanced pair in the histogram after embedding gets a bin-filling which corresponds to an original histogram. Due to the random selection of the coefficients, the remaining pairs can be corrected as well.

F3 does not directly manipulate bits but reduces the absolute value of a coefficient in case the LSB value does not match the value to be embedded, breaking the PoV structure in this way. 0-coefficients are not used, 1-coefficients are used (which causes the higher capacity as compred to Outguess). It has to be noted that coefficients with 1 and -1 value are reduced to 0 (“shrinkage”).

Steganographic Algorithms: F3 & F4

The receiver is not able to distinguish between unused 0-coefficients and shrinkage coefficients, which results in the procedure to embed the corresponding bit once again. This leads to higher share of even coefficients since for coefficients 1 and -1 nothing is changed if embedding a 1, but shrinkage accurs if embedding a 0, causing the next coefficient to be even again. One approach to counter this effect is to code messages in a way to balance this by using an appropiate share of 0/1. F4 solves this issue in a more elegant manner.



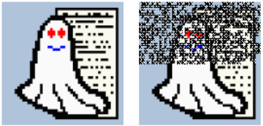


Negative coefficients are mapped to the inverse steganographic value: even negative coefficients represent an embedded 1, odd negative coefficients represent a 0, positive coefficients remain as usual. This simple idea delivers a histogram like it is desired, except for the excess of 0-coefficients.

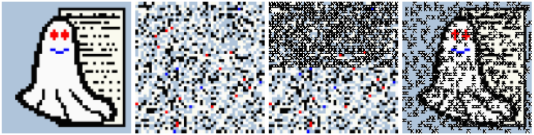
Steganographic Algorithms: F5 I

F5 improves F4 in two ways: permutative embedding and matrix embedding.

Classical sequential embedding changes the cover until the last data point manipulated:

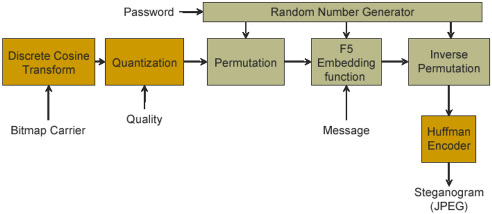


The simple idea permutes the data before embedding, embeds the data and performs back-permutation which distributed the manipulated coefficients uniformly across the image.



Steganographic Algorithms: F5 II

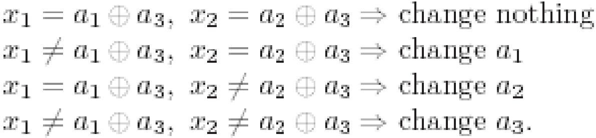
The graphic shows that back-permutation after embedding is done before inverse DCT, which explains the uniform distribution of the embedded data.



Matrix embedding reduces the required bit-manipulations per embedded stego-bit. Typically, 2 bits can be embedded per manipulation (since in 50% of all cases no manipulation needs to be conducted. Shrinkage reduces this valuie further, F5 matrix embedding is able to increase the efficiency to 3.8 embedded bits per manipulated bit.

Steganographic Algorithms: F5 III

Suppose we want to embed two bits x_1, x_2 at three admissible positions a_1, a_2, a_3 and the aim is to maximally manipulate a single position:

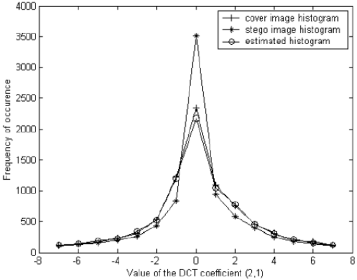


$$\begin{aligned}
x_1 = a_1 \oplus a_3,\ x_2 = a_2 \oplus a_3 &\Rightarrow \text{change nothing} \\
x_1 \neq a_1 \oplus a_3,\ x_2 = a_2 \oplus a_3 &\Rightarrow \text{change } a_1 \\
x_1 = a_1 \oplus a_3,\ x_2 \neq a_2 \oplus a_3 &\Rightarrow \text{change } a_2 \\
x_1 \neq a_1 \oplus a_3,\ x_2 \neq a_2 \oplus a_3 &\Rightarrow \text{change } a_3.
\end{aligned}$$

In the general case we have a codeword a with n bit positions to be changed for k bits of the message x. F is a hash function, which extracts k bits from the codeword, matrix embedding delivers a modified codeword a' for all x and a: $x = F(a')$ and the Hamming distance $d(a, a') \leq d_{max}$. This means that the codeword with n bits is changed at maximal d_{max} positions to embed k bits. In the example, the hash function is the relation among the a_i. In F5, $d_{max} = 1$ (Hadamard codewords are used).

Steganalysis: Calibration against F5

Calibration is a technique to derive properties of the cover image from the suspected stego image. In particular, the number of 0-coefficients is addressed (which is too high caused by shrinkage). The stego image is decompressed, cropped by 4 columns and recompressed using the same quantisation value. The figure shows how well histogram properties can be estimated.

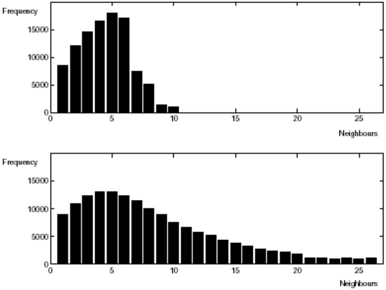


In case the quality is not too low, stego image is visually and in terms of PSNR similar to the cover image. The shift destroys the block structure of the manipulated quantised coefficients and some properties get very similar to the cover image. Besides cropping, other means of desynchronisation like rotation etc. can be used.

Steganalysis: Calibration against Outguess

Basis is the computation of a “blockiness” measure B, which computes pixel differences across horizontal and vertical block borders of adjacent pixel blocks. B increases monotonically with the number of flipped DCT LSBs. The first derivative (slope) of this relation is maximal for the cover image and is reduced for an image already containing a message.
At first, B of the decompressed stego image is computed: $B_S(0)$.
Subsequently, a message with maximal length is embedded and B of the result is determined: $B_S(1)$. Now, we apply calibration and recompression with identical quanitsation and compute B: $B(0)$. Again, we embed a maximal message and compute B: $B(1)$. And again: $B1(1)$.
Now we compute $S_0 = B(1) - B(0)$ as the slope of the original image ($p = 0$), $S_1 = B1(1) - B(1)$ as slope of an image with maximal message embedded and $S = B_S(1) - B_S(0)$ is between S_0 and S_1. The length of the message is $p = \frac{S_0 - S}{S_0 - S_1}$ (by applying linear interpolation $S = S_0 - p(S_0 - S_1)$). In case of $p = 0$, no message is embedded.

Color Embedding / Steganalysis



The steganographic algorithm “Hide” embeds in one of three colour channels, by increasing or decreasing the absolute value by 1 (instead of LSB flipping).

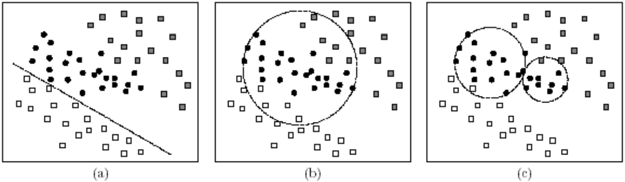
As a result it is clearly visible, that the number of “neighbouring” colours per colour pixel is higher as compared to natural images (e.g. 26 neighbours instead of 10; neighbours are pixels which only differ inone colour channel by one value). This can be detected statistically, of course.

Blind / Universal Steganalysis I

Steganographic techniques add noise, the histogrm gets smoother – which can be seen as a low-pass filter applied to the histogram. This observation can be used to compute a Fourier transform of the histogram (“histogram characteristic function”): Higher frequencies of stego images exhibit a lower energy compared to cover images. Ideally, techniques similar to calibration are desired in order to estimate the cover image. Since embedding is adding noise, denoising algorithms are used. For example the first k central moments $\sum_{i,j} |r[i,j] - mean(r)|^k$ of noise residuals r (original image minus denoised image), computed from the wavelet coefficients, are proposed as features.

Blind / Universal Steganalysis II

Farid et al [16] applies a similar idea: After computing the wavelet transform, mean, variance, skew, and kurtosis are computed. To model subband correlations, the identical values are computed from the prediction error of a linear predictor for each subband: The predictor is a weighted mean of adjacent coefficients of subbands of a different colour, of a different orientation, or a different scale – weights are computed in an error-minimisation procedure.



In subsequent work, a one-class SVM is used trained on cover images only. By doing this it is avoided to train with incomplete training-data, since we will easily miss certain stego techniques !

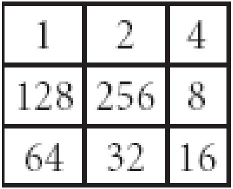
Blind / Universal Steganalysis III

Image quality measures (IQM) have also been suggested to differentiate stago images from cover images [4]. Based on an estimation of the cover image from the stego image (by denoising / Wiener filtering) various IQMs are computed from stego images, cover images, and their respective filtered versions.

Extensive statistical analysis shows that for different scenarios (e.g. active / passive attacker, robust WM, steganographic embedding), different IQM are suited to detect embedded data. For each scenario, a set of discriminative IQM can be identified for the values of which a regression on training data is computed. In the application case on test data, the class is chosen exhibiting the lowest distance to the regression result.

Blind / Universal Steganalysis IV

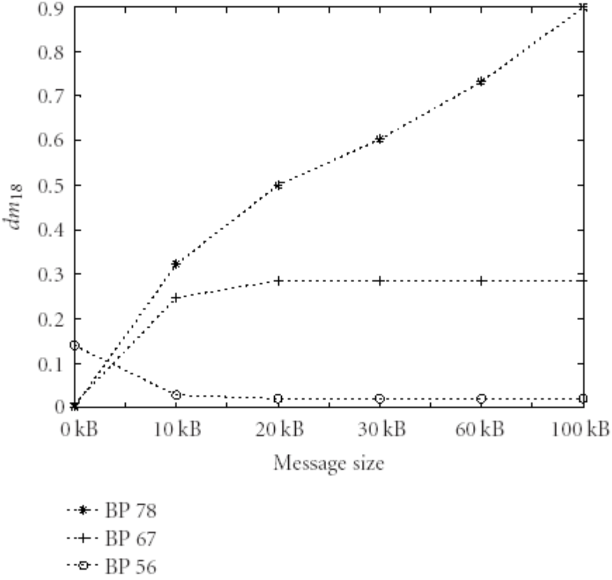
Binary similarity measure are used to detect the assumed decrease of correlation between the 7th and 8th bitplane in case of LSB embedding [3]. As an example local binary pattern (LBP) can be used. For each bitplane, a 512-bin histogram is established based on weighted neighbourhoods, the value at a pixel position is $S = \sum_{i=0}^{7} x_i 2^i$ where the weights 2^i are assigned to the neighbours as shown in the image. The entries in the n-th bin of the x-th bitplane are denoted as S_n^x. As an example for a measure used we give the Ojala Kullback-Leibler distance: $-\sum_{n=0}^{511} S_n^7 \log \frac{S_n^7}{S_n^8}$.



1	2	4
128	256	8
64	32	16

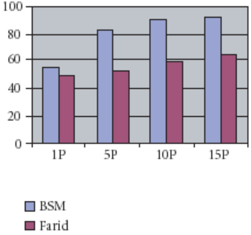
We notice a change in this measure for increasing message length under LSB embedding, other bitplanes do not exhibit this change.

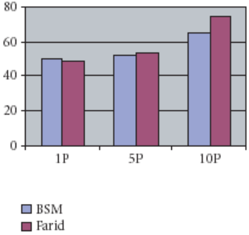
Blind / Universal Steganalysis V

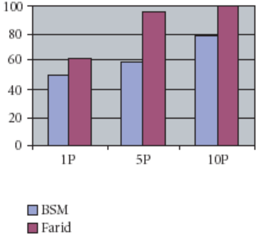


Blind / Universal Steganalysis VI

It is clearly visible, that for different embedding techniques different features are well suited for classification / detection. This results shows, that these techniques are not really entirely “universal”.

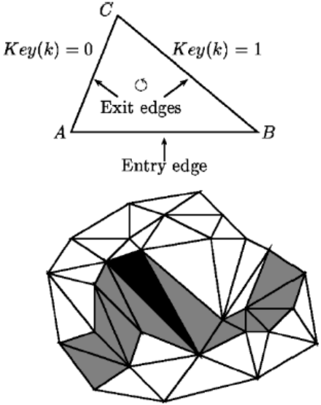






For binary similarity measures, a large set of different techiques can be employed, however, for any given dataset, the most appropriate one needs to be identified first. Thus, Farids technique seems to be more “universal”.

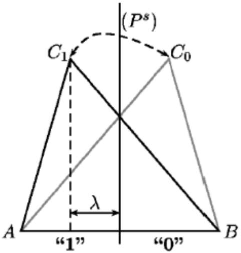
Steganography for 3D Meshes I

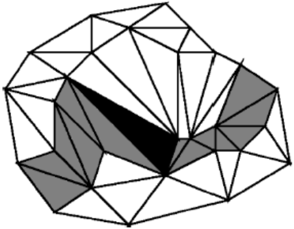


A proposed scheme consists of two stages: A list of triangles which contain the message (controlled by a PRNG), and a modification of the triangles according to the symbol to be embedded.

After the identification of a start triangle (based on specific geometric properties), the second triangle is determined according to the value of the PRNG, the exit edge of the current triangle is the entry edge of the subsequent one. In each triangle, a bit is embedded.

Steganography for 3D Meshes II

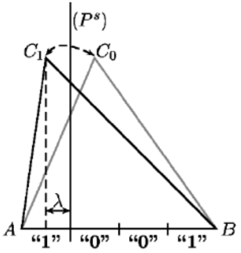


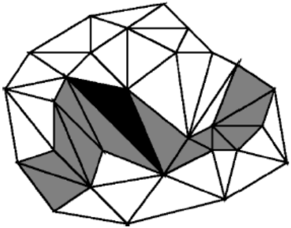


Each triangle has two conditions. The condition is defined by the position of the orthogonal projection of the triangle's vertex above the entry edge to the entry edge.

The entry edge is partitioned into two sections corresponding to the bits to be embedded. In case the projection is already in the correct section, no manipulation is required, otherwise the triangle's vertext is shifted to result in the correct projection. The distance of the border-line λ determines robustness and distortion of the technique.

Steganography for 3D Meshes III





In the example we display the option of using four sections instead of two, resulting in lower robustness but also lower impact on perceptability. For this technique we have reversability as an interesting property: In case λ as well as the "erasing key" (information which triangles have been modified) are kept, the embedding procedure can be reversed approximately.

Integrity Verification for Visual Data: Basics

Image processing and manipulation tools facilitate to change visual data without leaving perceptual traces. Thus, methods to reliably verify the integrity of these data are highly desirable. In classical cryptography, hash functions (MD-5, SHA-1,2,3), often used in a combination with public key crypto resulting in digital signature schemes (e.g. DSA) are used. Certain properties of media data indicate that specific or at least adapted techniques are required in this environment.

In classical crypto, minmal differences raise an alarm. This is not desired in media processing, since there are many manipulations which change the digital representation of the data but maintain the perceived content (e.g. format conversion, compression, filtering, ...). For this purpose, techniques have been developed which aim at verifying the integrity of perceivable content instead of the digital representation: semi-fragile WM and robust media hash functions.

Integrity Verification: Watermarks & Hash Functions

We distinguish between “fragile” WMs (which, similar to classical hash functions, indicate smallest changes to the visual content of the data) and “semi-fragile” WMs, which raise alarm in case of perceptually relevant content modifications. By analogy, robust hash functions also restrict their attention to perceptually relevant changes (this also motivates the notion of “perceptual hashes”).

Main advantages of WM as opposed to hashes is that these are integral part of the image data and in most cases, detected changes can also be localised. On the other hand, image data get modified by WM embedding (which might be reversible depending on the technique used).

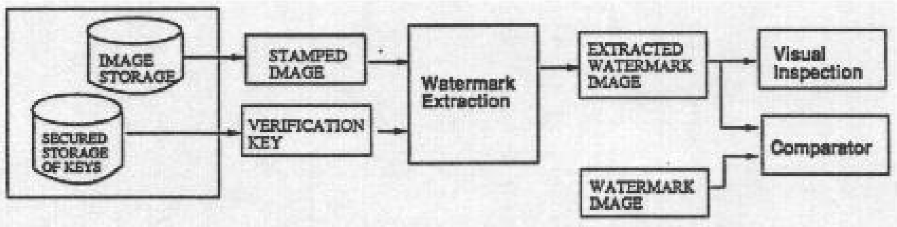
WMs and robust hashes can be combined: Hash values can be embedded into data by using WM technology. In this case, usually robust WM embedding techniques are used.

Fragile WM for Integrity Verification (IBM) I

The subsequent scheme has been developed by Yeung-Mintzer [18] at IBM Watson. For embedding, the cover image, a WM (binary image equally sized sa the cover image, eventually generated by circular repetition from a smaller WM) and a key is required. The key is stored in a protected database – only with this key the WM can be extracted and thus the integrity verified. The marked image can be published. For verification of the image content the key, the marked image and the WM are required. The original image is not needed (oblivious procedure). The extraction process delivers a binary image which is compared to the original WM. Any difference indicates (and localizes) a modification. Verification can be done manually (i.e. visually) or automated.

Fragile WM for Integrity Verification (IBM) II

vspace*1cm



Fragile WM for Integrity Verification (IBM) III

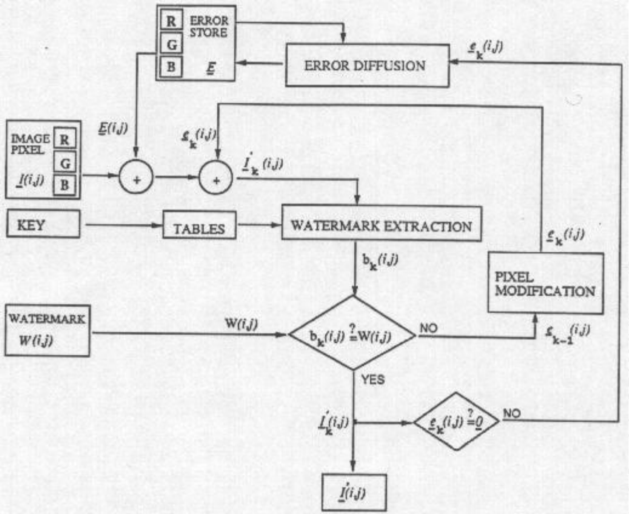
The following illustration shows the embedding process for a single pixel: W(i,j,) is the WM, I(i,j) the cover image, (i,j) the pixel location, $b_k(i,j)$ an extracted WM, $e_k(i,j)$ the difference between W and b_k and E(i,j) the diffused error for a specific colour channel.
The image is processed pixel by pixel in a defined scan order. The WM data is not explicitely embedded, but the image is systematically modified until the value of b_k matches that of W.
The following operations are conducted:

- Watermark Extraction: This function maps the colour values of the current pixel into a binary value. This function relies on a key (look-up tables), which can be generated randomly. The results of each colour channel are XORed.
- Pixel Modification: In each iteration one of the three colour channels is changed by +1 or -1 (as randomly as possible), this is iterated until the extraction process delivers the required bit.

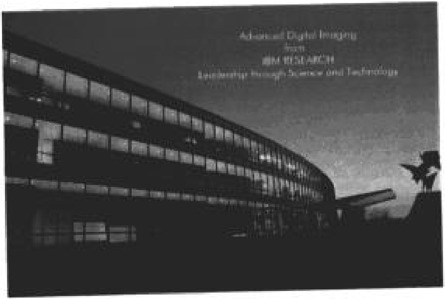
Fragile WM for Integrity Verification (IBM) IV

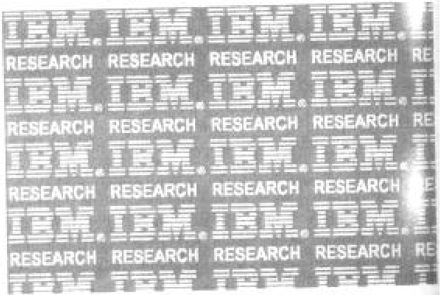
- Error Diffusion: WM embedding causes deviations form the original. The aim of this process is to maintain average pixel values in a neighbourhood to maintain colour impression. At initialisation E(i,j) = 0 for all (i,j). The current colour value is P(i,j) = I(i,j) + E(i,j), ie. the colour value of the original pixel plus the already performed error diffusion. I'(i,j) is the desired process output, such that W(i,j) = $b_k(i,j)$. The error to achieve this is $\delta(i,j)$ = P(i,j) - I'(i,j) – since the error is not desired, it is locally distributed bewteen the pixels right and below the current pixel (the half of $\delta(i,j)$ is added to each of them).

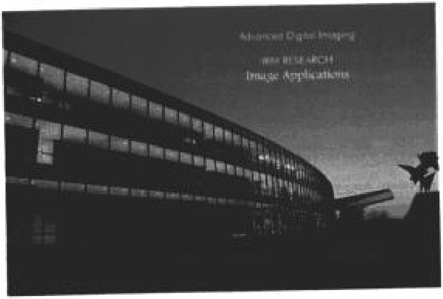
Fragile WM for Integrity Verification (IBM) V

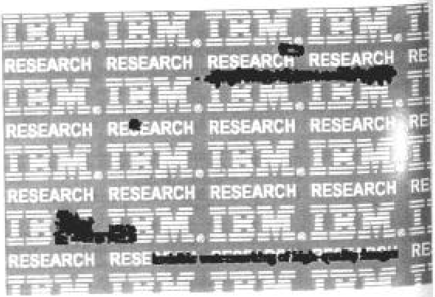


Fragile WM for Integrity Verification (IBM) VI









Fragile WM for Integrity Verification (IBM) VII

The algorithm has some nice properties:

- High speed.
- Highly sensitive against manipulations, not only the LSB is used.
- Extraction of the WM is only possible once the key is known.
- A potential attack tries to generate the key-tables if the WM is known or visually sensible (this is especially dangerous in case of gray-scale images, since their tables only have 256 entries in case of 8bpp). Thus, the WM should neither be public nor should it exhibit visible structures. This is a potential application case for chaotic mixing, where the WM does not exhibit any visible structure afterwards– the WM owner can reverse the mixing process after WM extraction.

Fragile WM for Integrity Verification (IBM): Attacks I

- The most effective attacks assume that identical keys and WMs have been used for several images. In this case we find for images u, v:

  $$W(i,j) = f_g(u(i,j)) = f_g(v(i,j)) \forall (i,j) \tag{7}$$

  From this equation we are able to identify groups of grey values, which are mapped to the same value by f. Within these groups pixel can be interchanged without changing the WM. In many cases, 90% of f can be reconstructed using 2 images only. A further assumption for this attack is the independence of f w.r.t.pixel position (i,j). Replacing f by $f(u(i,j),i,j)$, this attack is disabled.

Fragile WM for Integrity Verification (IBM): Attacks II

- Holliman-Memon (Collage) Attack: We assume to have a set of images $I_1, I_2, \ldots, I_n$ marked by the same key and WM. From these data we can synthetise an image J such that:

  $$\forall i, j J(i,j) = I_k(i,j) k \in 1, \ldots, n \tag{8}$$

  The quality of J depends on the number of images available. As before the appropriate counter measure is an image dependent function f (which leads to WM detection problems in the modified image), as an alternative f may involve more pixels (neighbours) besides the current one. The latter strategy worsens localisation of course.

Fragile WM for Integrity Verification (Alg. of Wong)

1. The image is partitioned into 8x8 pixel sized blocks Z_i in non-overlapping manner. Each block is processed further independently.
2. A binary WM is repeated periodically until the cover image size is used and partitioned in the same manner into A_i.
3. For each block Z_i, Z_i^* is the block obtained by deleting the LSBs. $H_i = Hash(Z_i^*)$.
4. $H_i' = H_i \oplus A_i$
5. H_i' is encrypted using a private key k, the result is the block signature $S_i = E_k(H_i')$.
6. S_i is inserted into the LSB of Z_i^*.

Fragile WM for Integrity Verification (Attacks against the Alg. of Wong)

- Collage attack: As before, assuming the attacker has access to several image protected by the same WM and key, a new image can be composed using blocks from the available images. This is also true for blocks within a single image, which can be moved freely. This attack is facilitated by the independence of the block signature from the block position and can be avoided by introducing position variance into the signature. Addtionally, besides Z_i itself neighbouring blocks could be used to compute the hash value H_i, again impacting localisation.
- Birthday attack: The scheme is restricted to hash values of 64 bit by construction due to the size of the blocks. A hash of this size can be attacked by searching for blocks with identical hash values. Again, larger blocks could be used, but this worsens localisation.

Semi-fragile WM for Integrity Verification I

Semi-fragile WM algorithms usually consist of two stages: The generation of WM information based on the cover image and a key and the embedding of these data into the cover image.
Two different strategies can be found in literature:

- Local Verification: Generation and embedding is done independently for single blocks. Localisation of modifications is done by verifying each single block.
- Global Verification: The WM describes global image properties and is also embedded into the entire image. Modifications are detected by comparing the extracted WM to the image. Often, the WM represents a miniaturised version of the image. The generation of the WM tries to capture significant image properties in a low number of bits (here robust hashes come into play !!). Besides an automated comparison, the WM can be verified visually. An additional feature is that modified image areas can at least be approximatively recovered using the WM informations. Embedding is done using robust WM techniques classically.

Semi-fragile WM for Integrity Verification II

The algorithm of von Lin, Podichuk und Delp [9] is an example for the first class of approaches:

- For each 8x8 pixel block a WM is generated. For this purpose, a key-dependent Gaussian distributed sequence of PR numbers is generatd: $C_1, \ldots C_{35}$. These are considered to be low frequency AC coefficient of a DCT block. The DC coefficient is set to zero, to avoid visible image changes. High frequency AC coefficients are not used due to their compression sensitivity.
- The IDCT of the block generates the WM W.
- The WMed block Y is generated, where σ controls the strength of the mark:

$$Y = X + \sigma W \quad (9)$$

Semi-fragile WM for Integrity Verification III

This scheme can be attacked as follows:

- In case an attacker has access to the original and the marked image, he/she can extract the WM and sign other images. This can be avoided by using image features as WM or by using image dependent WMs (when generating the coefficients).

Most local verification algorithms are sufficiently stable against compression and noise and have good localisation properties, however, geometrical transformations like rotation or arping destroy the block structure (and can be considered therefore as their “natural enemies”.

Robust Media Hash Functions I

Robust media hash function are used in different application domains, among them image verification and search in image databases. Different application fields require different algorithmic properties. However, there are also common requirements:

- Uniform distribution

$$Pr[H_K(X)] = \alpha] \approx \frac{1}{2^L}, \forall \alpha \in 0, 1^L. \tag{10}$$

- Pairwise independence in case of visually different images

$$Pr[H_K(X) = \alpha | H_K(Y) = \beta] \approx Pr[H_K(X)], \forall \alpha, \beta \in 0, 1^L. \tag{11}$$

- Invariance in case of visual similarity

$$Pr[H_K(X) = H_K(X')] \approx 1 \tag{12}$$

Robust Media Hash Functions II

Invariance in case of visual similarity results in a robust scheme. Similarity depends on semantic image content and the HVS and is not easy to quantify. Effective hash functions need to rely on image features being invariant to a wide class of transforms and representing perceived image content well.

Differences w.r.t. resistence against intentional modifications are seen depending on the intended application scenario:

- Integrity verification: It should be computationally infeasible to find an image X' being visually different to a given image X such that $H(X) = H(X')$. Visually different can mean any image manipulation, e.g. deleting or inserting objects.

Robust Media Hash Functions II

- CBIR (content based image retrieval): The construction of a visually similar image X' (to a given image X with $H(X) \neq H(X')$) should be computationally infasible.

Additionally it is often desired that the Hamming distance serves as a measure for the significance of the visual (dis)similarity. Due to the problems with systematic attacks exploiting this property this is not desirable when having security in mind. A further problem for the security of robust media hash functions is that they rely on deterministic image properties (e.g. pairs of DCT coefficients, mean values, median values) which facilitates a systematic manipulation of image content in many cases.

Integrity Verification Example A

Fridrich and Goljan [17] propose an algorithm which avoids publicly known image features. It expoints the fact that low frequency DCT coefficients are robust against many image manipulations and that their change leads to visible image artifacts.

1. Generate N random matrices $M^{(i)}$, the elements of which being uniformly distributed in [0..1], using a secret key K.
2. A lowpass filter is repeatedly applied to $M^{(i)}$ resulting in N smooth patterns $P^(i)$.
3. The mean of each pattern $P^(i)$ is substracted resulting in a 0 DC coefficient.
4. The hashbits b_i result from projecting image blocks to the patterns $P^{(i)}$ with Th being a constant threshold:

$$if|B.P^{(i)}| < Th\, b_i = 0 \quad if|B.P^{(i)}| \geq Th\, b_i = 1 \qquad (13)$$

Integrity Verification Example B

A further proposal is given by Venkatesan et al. (Microsoft Research) [49]:

1. The image is wavelet transformed.
2. For each subband we compute a feature vector F_i. For this purpose, each subband is randomly subdivided and for each region a statistical descriptor is computed (mean for the LL band, otherwise variance).
3. The float values of each F_i are mapped to $0, \ldots, 7$ by randomised rounding. The result is the preliminary hashstring H_p.
4. For making the hash shorter and more robust, a Reed-Muller error-correcting code is applied to H_p in decoding mode, followed by a linear code.

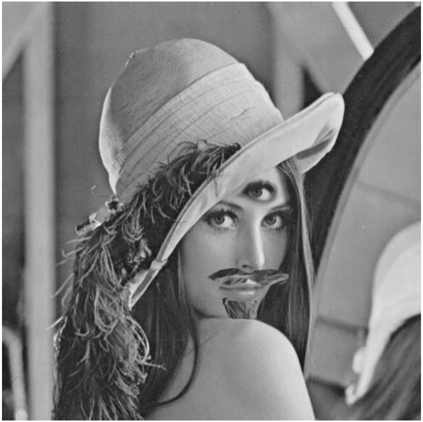
Attack against Integrity Verification Example B

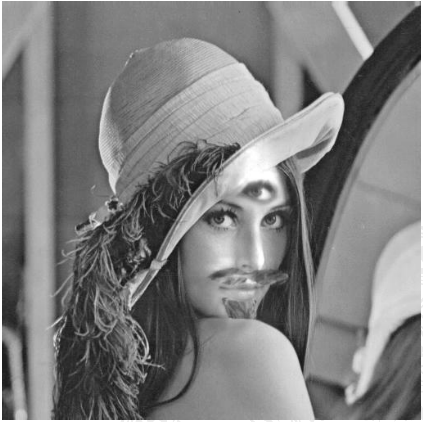
Meixner & Uhl [26] show how to attack the proposed scheme:

1. For a given image I the manipulated image F is constructed, both are wavelet transformed.
2. For I and F, identical partitions $I_1, \ldots, I_n$ and $F_1, \ldots, F_n$ are constructed.
3. Depending on the type of subband: $\forall$ regions i mean or variance in F_i is manipulated to match the value in I_i a
4. F is invers wavelet transformed to obtain the desired (faked) image.

It turns out that the random subdivision does not prevent attacks, since for arbitrary large n the exact partitioning scheme does not need to be known. The extent of the manipulation of course influences quality as well as the similarity of the hash values.

Attack against Integrity Verification Example B – Visualisation





The faked image exhibits a Hamming distance of 0.02 to the original, compared to a distance of 0.1 at JPEG quality 90.

Robust Media Hash Functions: CBIR Example A

Mihcak und Venkatesan (Microsoft Research) [50] propose the following procedure for this application scenario:

- Apply an L level wavelet transform to the image X. X_A is the resulting approximation subband.
- X_A is converted to binary M by thresholding (T is the median of X_A):

$$M_1(i,j) = \begin{cases} 1 \; X_A(i,j) \geq T \\ 0 \; otherwise \end{cases} \tag{14}$$

- $S_{[m,n],p}(A)$ is a statistical order filter, with: $S_{[m,n],p}(A) = B$, where $\forall i, j B(i,j)$ is the p-th element of the ascendingly sorted set $A(i',j') : i' \in i-m, \ldots, i+m, j' \in j-n, \ldots, j+n$, and f is a shift invarinat filter function.

Robust Media Hash Functions: CBIR Example A contd.)

- $M_2 = S_{[m,n],p}(M_1)$
- $M_3 = A.M_2$, $M_4 = f(M_3)$
- The binary image M_5 is generated from M_4 by thresholding
- In case iteration count $> C$ the algorithm stops, otherwise $D(M_1, M_5) < \epsilon$ is verified and in case of *false* the procedure re-iterates the computation of M_2.
- $H = M_5$

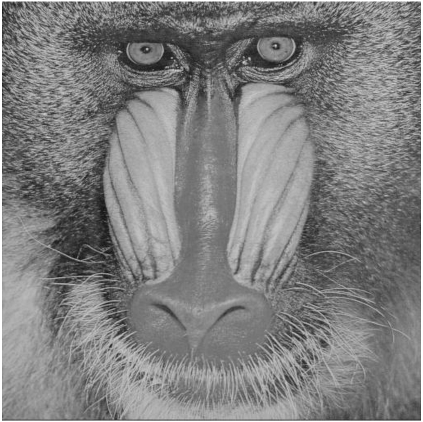
The obtain key-dependence, the algorithm is applied to randomly generated rectangles R_i and the resulting hash bits are concatenated in random order, subsequently a subset is selected. This subset is the final hash value $H_K(X)$.

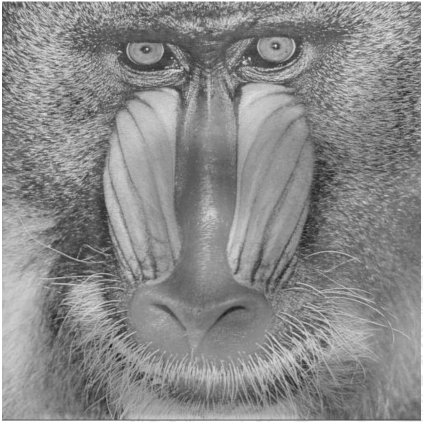
Attacking CBIR Example A

An attack against a CBIR hash tries to modify an image such that the resulting hash is as different as possible, while maintaining the visual semantics. The best attack point (as demonstrated by Meixner and Uhl [27]) is when the first binary image is generated by thresholding, where the mean is used to generate an equal number of black and white pixels.

For the attack, the image is DWT transformed and the median of the LL subband is determined. Subsequently, coefficients close to the median are changed (which does not change much in terms of perception, but has significant impact to the thresholded image. The faked image is generated by applying IDWT.

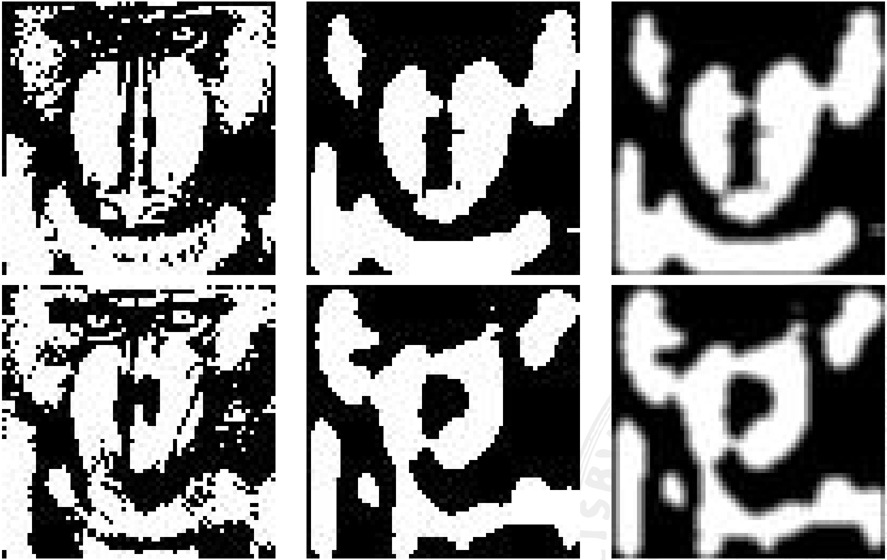
Attacking CBIR Example A: Visualisation





The faked image has a Hamming distance of 0.35 to the original, while "Truck" has a distance of 0.38 to the original "Baboon".

Attacking CBIR Example A: Further Visualisation



Outline

1. Formalia
2. Introduction & Motivation
3. Encryption of Visual Data
   - Basics
   - Pay TV Systems
   - DVD Security
   - BLU RAY Security
   - Partial Encryption Schemes
4. Information Hiding & Watermarking
   - Robust Watermarking
   - Steganography
   - Integrity Verification for Visual Data
5. **Literature**

Literature I

I. Agi and L. Gong.
An empirical study of secure MPEG video transmissions.
In *ISOC Symposium on Network and Distributed Systems Security*, pages 137–144, San Diego, California, 1996.

Fadi Almasalha, Nikita Agarwal, and Ashfaq Khokhar.
Secure multimedia transmission over RTP.
In *Proceedings of the Eighth IEEE International Symposium on Multimedia (ISM'08)*, pages 404–411, Berkeley, CA, USA, December 2008. IEEE Computer Society.

Ismail Avcibas, Nasir Memon, and Bülent Sankur.
Image steganalysis with binary similarity measures.
In *Proceedings of the IEEE International Conference on Image Processing (ICIP'02)*, Rochester, USA, September 2002.

Ismail Avcibas, Nasir Memon, and Bülent Sankur.
Steganalysis using image quality metrics.
IEEE Transactions on Image Processing, 12(2):221–229, February 2003.

H. Cheng and X. Li.
On the application of image decomposition to image compression and encryption.
In P. Horster, editor, *Communications and Multimedia Security II, IFIP TC6/TC11 Second Joint Working Conference on Communications and Multimedia Security, CMS '96*, pages 116–127, Essen, Germany, September 1996. Chapman & Hall.

H. Cheng and X. Li.
Partial encryption of compressed images and videos.
IEEE Transactions on Signal Processing, 48(8):2439–2451, 2000.

Ingemar J. Cox, Matthew L. Miller, and Jeffrey A. Bloom.
Digital Watermarking.
Morgan Kaufmann, 2002.

Literature II

Scott A. Craver, Nasir Memon, Boon-Lock Yeo, and Minerva M. Yeung.
Can invisible watermarks resolve rightful ownerships?
In Ishwar K. Sethi and Ramesh C. Jain, editors, *Proceedings of SPIE, Storage and Retrieval for Image and Video Databases*, volume 3022, pages 310–321, San Jose, CA, USA, July 1996.

Edward J. Delp, Christine I. Podilchuk, and Eugene T. Lin.
Detection of image alterations using semi-fragile watermarks.
In Ping Wah Wong and Edward J. Delp, editors, *Proceedings of IS&T/SPIE's 12th Annual Symposium, Electronic Imaging 2000: Security and Watermarking of Multimedia Content II*, volume 3971, San Jose, CA, USA, January 2000.

W. M. Dietl and A. Uhl.
Robustness against unauthorized watermark removal attacks via key-dependent wavelet packet subband structures.
In *Proceedings of the IEEE International Conference on Multimedia and Expo, ICME '04*, Taipei, Taiwan, June 2004.

Werner Dietl, Peter Meerwald, and Andreas Uhl.
Protection of wavelet-based watermarking systems using filter parametrization.
Signal Processing (Special Issue on Security of Data Hiding Technologies), 83(10):2095–2116, October 2003.

Jana Dittmann, editor.
Digitale Wasserzeichen: Grundlagen, Verfahren, Anwendungsgebiete.
Springer Verlag, 2000.

Jana Dittmann and Ralf Steinmetz.
Enabling technology for the trading of MPEG-encoded video.
In *Information Security and Privacy: Second Australasian Conference, ACISP '97*, volume 1270, pages 314–324, July 1997.

Literature III

Dominik Engel, Rade Kutil, and Andreas Uhl.
A symbolic transform attack on lightweight encryption based on wavelet filter parameterization.
In *Proceedings of ACM Multimedia and Security Workshop, MM-SEC '06*, pages 202–207, Geneva, Switzerland, September 2006.

Dominik Engel, Thomas Stütz, and Andreas Uhl.
Assessing JPEG2000 encryption with key-dependent wavelet packets.
EURASIP Journal on Information Security, 2012(2), 2012.

Hany Farid and Siwei Lyu.
Detecting hidden messages using higher-order statistics and support vector machines.
In *Proceedings of the 5th International Workshop on Information Hiding, IH '02*, Lecture Notes in Computer Science, Noordwijkerhout, The Netherlands, October 2002. Springer-Verlag.

Jiri Fridrich.
Visual hash for oblivious watermarking.
In Ping Wah Wong and Edward J. Delp, editors, *Proceedings of IS&T/SPIE's 12th Annual Symposium, Electronic Imaging 2000: Security and Watermarking of Multimedia Content II*, volume 3971, San Jose, CA, USA, January 2000.

Jiri Fridrich, Miroslav Goljan, and Nasir Memon.
Further attacks on yeung-mintzer fragile watermarking schemes.
In Ping Wah Wong and Edward J. Delp, editors, *Proceedings of IS&T/SPIE's 12th Annual Symposium, Electronic Imaging 2000: Security and Watermarking of Multimedia Content II*, volume 3971, San Jose, CA, USA, January 2000.

Raphaël Grosbois, Pierre Gerbelot, and Touradj Ebrahimi.
Authentication and access control in the JPEG2000 compressed domain.
In A.G. Tescher, editor, *Applications of Digital Image Processing XXIV*, volume 4472 of *Proceedings of SPIE*, pages 95–104, San Diego, CA, USA, July 2001.

Literature IV

Michael Gschwandtner.
Efficient protection and access control of 3d geometry data.
Master's thesis, Department of Scientific Computing, University of Salzburg, Austria, March 2009.

Michael Gschwandtner and Andreas Uhl.
Toward DRM for 3D geometry data.
In Edward J. Delp III, Ping Wah Wong, Jana Dittmann, and Nasir D. Memon, editors, *Proceedings of SPIE, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, volume 6819, page 68190V ff., San Jose, CA, USA, January 2008. SPIE.

Neil F. Johnson, Zoran Duric, and Sushil Jajodia.
Information Hiding: Steganography and Watermarking - Attacks and Countermeasures.
Kluwer Academic Publishers, 2000.

Stefan Katzenbeisser and Fabien A. P. Petitcolas.
Information Hiding Techniques for Steganography and Digital Watermarking.
Artech House, December 1999.

Eckhard Koch and Jian Zhao.
Towards robust and hidden image copyright labeling.
In *Proceedings of the IEEE International Workshop on Nonlinear Signal and Image Processing*, pages 452–455, Marmaras, Greece, June 1995.

Thomas Kunkelmann.
Applying encryption to video communication.
In *Proceedings of the Multimedia and Security Workshop at ACM Multimedia '98*, pages 41–47, Bristol, England, September 1998.

Literature V

Albert Meixner and Andreas Uhl.

Analysis of a wavelet-based robust hash algorithm.

In Edward J. Delp and Ping W. Wong, editors, *Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306 of *Proceedings of SPIE*, pages 772–783, San Jose, CA, USA, January 2004. SPIE.

Albert Meixner and Andreas Uhl.

Robustness and security of a wavelet-based CBIR hashing algorithm.

In *Proceedings of ACM Multimedia and Security Workshop, MM-SEC '06*, pages 140–145, Geneva, Switzerland, September 2006.

R. Norcen, M. Podesser, A. Pommer, H.-P. Schmidt, and A. Uhl.

Confidential storage and transmission of medical image data.

Computers in Biology and Medicine, 33(3):277 – 292, 2003.

M. Podesser, H.-P. Schmidt, and A. Uhl.

Selective bitplane encryption for secure transmission of image data in mobile environments.

In *CD-ROM Proceedings of the 5th IEEE Nordic Signal Processing Symposium (NORSIG 2002)*, Tromso-Trondheim, Norway, October 2002. IEEE Norway Section.
file cr1037.pdf.

A. Pommer and A. Uhl.

Wavelet packet methods for multimedia compression and encryption.

In *Proceedings of the 2001 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 1–4, Victoria, Canada, August 2001. IEEE Signal Processing Society.

A. Pommer and A. Uhl.

Application scenarios for selective encryption of visual data.

In J. Dittmann, J. Fridrich, and P. Wohlmacher, editors, *Multimedia and Security Workshop, ACM Multimedia*, pages 71–74, Juan-les-Pins, France, December 2002.

Literature VI

A. Pommer and A. Uhl.
Selective encryption of wavelet packet subband structures for obscured transmission of visual data.
In *Proceedings of the 3rd IEEE Benelux Signal Processing Symposium (SPS 2002)*, pages 25–28, Leuven, Belgium, March 2002. IEEE Benelux Signal Processing Chapter.

A. Pommer and A. Uhl.
Selective encryption of wavelet packet subband structures for secure transmission of visual data.
In J. Dittmann, J. Fridrich, and P. Wohlmacher, editors, *Multimedia and Security Workshop, ACM Multimedia*, pages 67–70, Juan-les-Pins, France, December 2002.

A. Pommer and A. Uhl.
Selective encryption of wavelet-packet encoded image data — efficiency and security.
ACM Multimedia Systems (Special issue on Multimedia Security), 9(3):279–287, 2003.

A. Pommer and A. Uhl.
Lightweight protection of visual data using high-dimensional wavelet parametrization.
In F. Roli and S. Vitulano, editors, *Image Analysis and Processing - ICIAP 2005*, volume 3617 of *Lecture Notes on Computer Science*, pages 645–652, Cagliari, Italy, September 2005. Springer-Verlag.

Lintian Qiao and Klara Nahrstedt.
Comparison of MPEG encryption algorithms.
International Journal on Computers and Graphics (Special Issue on Data Security in Image Communication and Networks), 22(3):437–444, 1998.

Josef Scharinger.
Robust watermark generation for multimedia copyright protection.
In *Proceedings of IWSSIP '99*, pages 177–180, Bratislava, Slovakia, 1999.

Literature VII

J. Schneid and S. Pittner.
On the parametrization of the coefficients of dilation equations for compactly supported wavelets.
Computing, 51:165–173, May 1993.

C. Shi and B. Bhargava.
A fast MPEG video encryption algorithm.
In *Proceedings of the Sixth ACM International Multimedia Conference*, pages 81–88, Bristol, UK, September 1998.

S.U. Shin, K.S. Sim, and K.H. Rhee.
A secrecy scheme for MPEG video data using the joint of compression and encryption.
In *Proceedings of the 1999 Information Security Workshop (ISW'99)*, volume 1729 of *Lecture Notes on Computer Science*, pages 191–201, Kuala Lumpur, November 1999. Springer-Verlag.

Champskud J. Skrepth and Andreas Uhl.
Selective encryption of visual data: Classification of application scenarios and comparison of techniques for lossless environments.
In B. Jerman-Blazic and T. Klobucar, editors, *Advanced Communications and Multimedia Security, IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security, CMS '02*, pages 213 – 226, Portoroz, Slovenia, September 2002. Kluver Academic Publishing.

G. Spanos and T. Maples.
Performance study of a selective encryption scheme for the security of networked real-time video.
In *Proceedings of the 4th International Conference on Computer Communications and Networks (ICCCN'95)*, Las Vegas, NV, 1995.

Thomas Stütz and Andreas Uhl.
On format-compliant iterative encryption of JPEG2000.
In *Proceedings of the Eighth IEEE International Symposium on Multimedia (ISM'06)*, pages 985–990, San Diego, CA, USA, December 2006. IEEE Computer Society.

Literature VIII

Thomas Stütz and Andreas Uhl.
(In)secure multimedia transmission over RTP.
In *Proceedings of the 18th European Signal Processing Conference, EUSIPCO '10*, Aarlborg, Danmark, August 2010. EURASIP.

L. Tang.
Methods for encrypting and decrypting MPEG video data efficiently.
In *Proceedings of the ACM Multimedia 1996*, pages 219–229, Boston, USA, November 1996.

Ali Saman Tosun and Wu chi Feng.
On error preserving encryption algorithms for wireless video transmission.
In *Proceedings of the ninth ACM Multimedia Conference 2001*, pages 302–307, Ottawa, Canada, October 2001.

T. Uehara and R. Safavi-Naini.
Chosen DCT coefficients attack on MPEG encryption schemes.
In *Proceedings of the 2000 IEEE Pacific Rim Conference on Multimedia*, pages 316–319, Sydney, December 2000. IEEE Signal Processing Society.

T. Uehara, R. Safavi-Naini, and P. Ogunbona.
Securing wavelet compression with random permutations.
In *Proceedings of the 2000 IEEE Pacific Rim Conference on Multimedia*, pages 332–335, Sydney, December 2000. IEEE Signal Processing Society.

Ramarathnam Venkatesan, S.-M. Koon, Mariusz H. Jakubowski, and Pierre Moulin.
Robust image hashing.
In *Proceedings of the IEEE International Conference on Image Processing (ICIP'00)*, Vancouver, Canada, September 2000.

Literature IX

Ramarathnam Venkatesan and M. Kivanc Mihcak.
New iterative geometric methods for robust perceptual image hashing.
In *Proceedings of the Workshop on Security and Privacy in Digital Rights Management 2001*, Philadelphia, PA, USA, November 2001.

Jiangtao Wen, Mike Severa, Wenjun Zeng, Max Luttrell, and Weiyin Jin.
A format-compliant configurable encryption framework for access control of multimedia.
In *Proceedings of the IEEE Workshop on Multimedia Signal Processing, MMSP '01*, pages 435–440, Cannes, France, October 2001.

Jiangtao Wen, Mike Severa, Wenjun Zeng, Max Luttrell, and Weiyin Jin.
A format-compliant configurable encryption framework for access control of video.
IEEE Transactions on Circuits and Systems for Video Technology, 12(6):545–557, June 2002.

Tsung-Li Wu and S. Felix Wu.
Selective encryption and watermarking of MPEG video (extended abstract).
In Hamid R. Arabnia, editor, *Proceedings of the International Conference on Image Science, Systems, and Technology, CISST '97*, Las Vegas, USA, February 1997.

Wenjun Zeng and Shawmin Lei.
Efficient frequency domain video scrambling for content access control.
In *Proceedings of the seventh ACM International Multimedia Conference 1999*, pages 285–293, Orlando, FL, USA, November 1999.