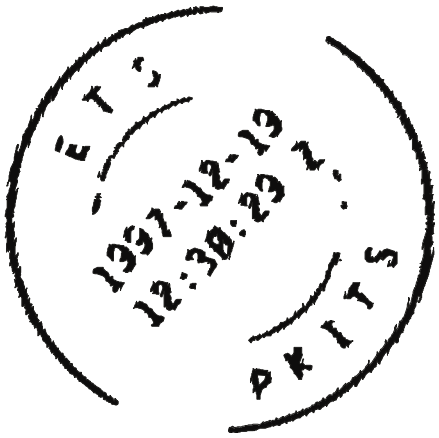


PKITS
Public Key Infrastructure with Time-Stamping Authority

ETS PROJECT: 23.192

Deliverable D6
Time-Stamping Service
Functional Specification and Protocols
for Multimedia Information

Produced by: ***FNMT***
Date of issue: ***30th August 1998***
Revision Number: 16

TABLE OF CONTENTS

GLOSSARY OF TERMS

Specifications:	
SHALL	Essential requirement. A requirement must be fulfilled or a feature implemented wherever this term occurs. The designer is requested, however, to indicate if one or more “shall requirements” would increase the cost or time unreasonably in relation to the total cost or design cost, in which case the specification may have to be revised.
SHOULD	Important requirement. Shall be implemented without or with minimum extra cost. Valid reasons in particular circumstances may allow ignoring such requirements.
MAY	Optional requirement. From case to case, it should be decided whether implementing it or not, in any case without exceeding the budget planned for the related activity.

Technical:	
BPP	Bits Per Pixel
bps	Bits Per Second
Bps	Bytes Per Second
EDI	Electronic Data Interchange
ETS	European Trusted Systems
FPS	Frames per second
DCT	Discrete Cosine Transform
HDTV	High Definition TV
JPEG	Joint Photograph Expert Group
MIME	Multipurpose Internet Mail Extensions
MPEG	Moving Picture Expert Group
PCM	Pulse Code Modulation
PKI	Public Key Infrastructure
TS	Time-Stamping
TSA	Time-Stamping Authority
TSS	Time-Stamping Service
TTP	Trusted Third Party

1 EXECUTIVE SUMMARY

This document covers the provision of time-stamping services for multimedia information: video, images, and audio, either in isolation or combined into streams. This deliverable extends the architecture and the protocols identified in PKITS deliverables D3 and D4 to apply to multimedia objects.

The document is organised as follows. Section 3 surveys techniques currently used to encode, transform, and transfer multimedia information; it does not cover the extremely rich variety of formats and means, but those that have become standards (either *de iure* or *de facto*), and are meaningful for time-stamping. Section 4 describes protocols to time-stamp multimedia information, in a comparative way, highlighting the pros and cons of each approach to help actual election in practice. Section 5 covers service requirements from the points of view of the service users (requester and verifier) and the providing TSA (software and hardware). Section 6 states the framework for an application programming interface that supports the required operations; it uses the JavaMedia framework as a concrete foundation. Section 7 presents a number of usage scenarios to exercise the previously presented protocols and techniques and analyse to which extend they apply to actual usage scenarios.

Appendixes incorporate background knowledge needed for the service: cryptographic primitives, and a summary of watermarking techniques.

1.1 WHAT IS "MULTIMEDIA INFORMATION"?

Multimedia information covers currently audio, images and video. This kind of information poses its own requirements:

- means to deal comfortably with high volumes of data: what is the right size to sign? How often shall the TSA be requested to put a time stamp? Local time stamping may relax communication problems, and introduces the requirement for periodic [re-]synchronisation exchanges
- means to provide economic stamping. Hashing and signing with a public key each one of the frames of a real-time video sequence may be technically unfeasible. Block stamping becomes a must.
- means to address information formats: there are many coding techniques, some of which work on content differences (e.g. a master frame is sent, and then a series of updates that only cover incremental differences with respect to the master frame)
- means to address information processing: images are often processed either for storage, for transmission, or just for convenience of use. Some transformations are reversible, but others are not.
- ordering is relevant in multimedia: a sequence of images must not be altered, and ordering should be protected against criminal manipulation
- sometimes, it is even more important the serialisation (relative ordering) that the absolute time
- multimedia information opens the opportunity to mix together the data and the time stamp; watermarking techniques are under development (mostly related to copyright issues) to provide survivability of associated information. These techniques could be exploited to provide evidence through transformations that alter the original material, but do not destroy the semantics that was to be notarised.
- multimedia information is often manipulated in such a way that human value is respected despite some quality loss, while current hashing techniques used to characterise data are bit sensitive (a single bit change makes the digest useless). More tolerant hashing algorithms would be interesting to provide a smooth transition from strong evidence to weaker confidence

The new opportunities to match TTP services to multimedia objects must take into consideration the large number of storage and transmission conventions (data encoding and exchange protocols). Furthermore, new formats must be foreseen as hypertext pervades the information highways. Therefore, a requirement oriented approach is in place, asking format inventors to implement the required services as feasible with each proposal. That is, to define **application level interfaces** for the provision of time stamping services.

1.2 PROJECT ROADMAP

PKITS structures its technical contributions into a collection of deliverables:

D3. Architecture of Time-Stamping Service and Scenarios of Use: Services and Features
D4. Time-Stamping Service Functional Specification and Protocols for Unstructured Data
D5. TSS Functional Specification and Protocols for EDI Messages and Interchanges
D6. Time-Stamping Service Functional Specification and Protocols for Multimedia Information

D3 studies the definition of the service, identifies applicable protocols, and establishes a working framework. D4 focuses on unstructured documents, that is those documents whose internal structure is unknown to the TSA, and it cannot benefit from any knowledge, nor deal with separate fields in any sensible manner. D5 and D6 consider those cases where there is a knowledge of the syntax and semantics of the documents: D5 studies the case of EDI messages, that are strictly formalised, and where there are fields specifically designed for holding time-stamping information; D6 studies multimedia information where there is knowledge of the audio and video stream structure. D5 and D6 propose mechanisms for embedding time-stamping information into already standardised information structures. D4, dealing with unstructured data, proposes an appendix model to wrap time-stamping information to raw data.

2 REFERENCES

[BeGru+96] W. Bender, D. Gruhl, N. Morimoto, A. Lu, "Techniques for Data Hiding", IBM Systems Journal, Vol. 35, Nos. 3 & 4, 1996.

[BossGov+96] A. Bosselaers, R. Govaerts, J. Vandewalle, "Fash hashing on the Pentium", Advances in Cryptology, Proceedings Crypto '96, LNCS 1109, Springer-Verlag, 1996. Also see the short note "Even faster hashing on the Pentium", presented in Eurocrypt '97.

[CoxKil+96] I. J. Cox, J. Kilian, T. Leighton, T. Shamoon, "A Secure, Robust Watermark for Multimedia", Workshop on Information Hiding, Newton Institute, Univ,. of Cambridge, May 1996.

[CoxKil+97] I.J. Cox, J. Kilian, T. Leighton, T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia,", *IEEE Trans. on Image Processing, 6, 12,* 1997.

[EvGoMi94] S. Even, O. Goldreich, S. Micali, "On-Line/Off-Line Digital Signatures", US Patent No 5,016,274..

[GennRoh97] R. Gennaro, P. Rohatgi, "How to Sign Digital Streams", Proceedings of CRYPTO' 97, July 1997.

[GiHar97] F Hartung, B Girod, "Watermarking of MPEG-2 encoded video without decoding and re-encoding", *Multimedia Computing and Networking* 1997, published as SPIE v 3020, 1997.

[ISO92a] ISO/IEC 10918 (JPEG), "Information Technology – Digital Compression and Coding of Continuous Tone Images", July 1992.

[ISO92b] ISO/IEC 11172 (MPEG-1), "Information Technology –Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1,5 Mbit/s", ISO/IEC JTC1/SC29/WG11**,** October 1992.

[ISO92c] ISO/IEC 13818 (MPEG-2), "Information Technology –Generic Coding of Moving Pictures and Associated Audio Information", ISO/IEC JTC1/SC29/WG11**,** October 1992.

[ISO95] MPEG-4 Proposal Package Description (PPD) – Revision 2, ISO/IEC JTC1/SC29/WG11**,** March 1995. The formal ISO/IEC designation for this standard (MPEG-4 is to be released in October 1998 and will be an International Standard in December 1998) will be ISO/IEC 14496.

[ITU93] ITU-T (CCITT) Specialist Group on Coding for Visual Telephony, "Recommendation H.261 – Video Codec for Audiovisual Services at p x 64 kbit/s", Mar. 1993.

[ITU95a] ITU-T (CCITT) Specialist Group on Coding for Visual Telephony, "Recommendation H.263 – Video Codec for Low Bitrate Communication", October 1995.

[ITU95b] ITU-T (CCITT) Specialist Group on Coding for Visual Telephony, "Recommendation H.324 –Terminal for Low Bitrate Multimedia Communication", November 1995.

[Kurak+92] C. Kurak, J. McHugh, "A Cautionary Note on Image Downgrading", Proceedings of the 8th Annual Computer Security Applications Conference, 1992.

[Menez+97] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1997.

[MiBra+97] F. Mintzer, G. W. Braudaway, M. M. Yeung, "Effective and Ineffective Digital Watermarks", IBM Research Report RC-20933, July 1997.

[Pitas96] I. Pitas, "A Method for Signature Casting on Digital Images", International Conference on Image Processing, Lausanne (Switzerland), September 1996.

[PKITS D3] Architecture of Time-Stamping Service and Scenarios of Use: Service and Features, Deliverable D3 of ETS Project 23.192 Public Key Infrastructure

with Time-Stamping Authority, May, 1998.

[PKITS D4] Time-Stamping Service Functional Specification and Protocols for Unstructured Data, Deliverable D4 of ETS Project 23.192 Public Key Infrastructure with Time-Stamping Authority, July, 1998.

[RFC 2045] N. Freed, N. Borenstein , "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", Innosoft, First Virtual Holdings, November 1996.

[RFC 2046] N. Freed, N. Borenstein , "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", Innosoft, First Virtual Holdings, November 1996.

[RFC 2047] K. Moore, "Multipurpose Internet Mail Extensions (MIME) Part Three: Message Header Extensions for Non-ASCII Text", University of Tennessee, November 1996.

[RFC 2048] N. Freed, J. Klensin, and J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", Innosoft, MCI, ISI, November 1996.

[RFC 2049] N. Freed, N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples" , Innosoft, First Virtual Holdings, November 1996.

[RFC 2311] S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, L. Repka, "S/MIME Version 2 Message Specification", March 1998.

[vanSch+94] R. G. van Schyndel, A. Z. Tirkel, C. F. Osborne, "A Digital Watermark", International Conference on Image Processing, Austin (Texas, USA), 1994.

[Weidai] Crypt++ Internet Home Page: http://www.eskimo.com/~weidai

3 MULTIMEDIA INFORMATION

The term "multimedia" refers to the use of a variety of media types such as video, audio, text, graphics, 3D animations, etc., together in one digital document that acts as container. The relationships of the different media types can be of significant complexity, such as:

- Synchronisation between the different streams, e.g. in order to ensure that the sound is displayed to the user in sync with the audio and text information.
- Presentation information that will contain the rules to follow when displaying the multimedia document to the user.
- Several alternative data streams corresponding to a same media type can be contained in a multimedia document, for example with different language versions of sound and text; the application should present the appropriate one.
- Although inherently sequential in nature, the multimedia streams must enable the user to access the multimedia document randomly, and in some cases it must enable the user to present the information at a rate different form the one used initially.
- When a multimedia document overloads the CPU, the system should be able to present the information discarding some media samples; this process can become quite complex, since some media streams affect to the user perception of quality in a greater extent than others.

Some important characteristics of multimedia documents are:

1. Representing multimedia information usually involves huge amounts of information. The sample sizes are usually so large that compression algorithms are used to reduce them. Although these compression algorithms often consume a large amount of memory and CPU cycles, the compression process usually results in an overall benefit, since the rest of the media manipulations will take place on smaller amounts of information.

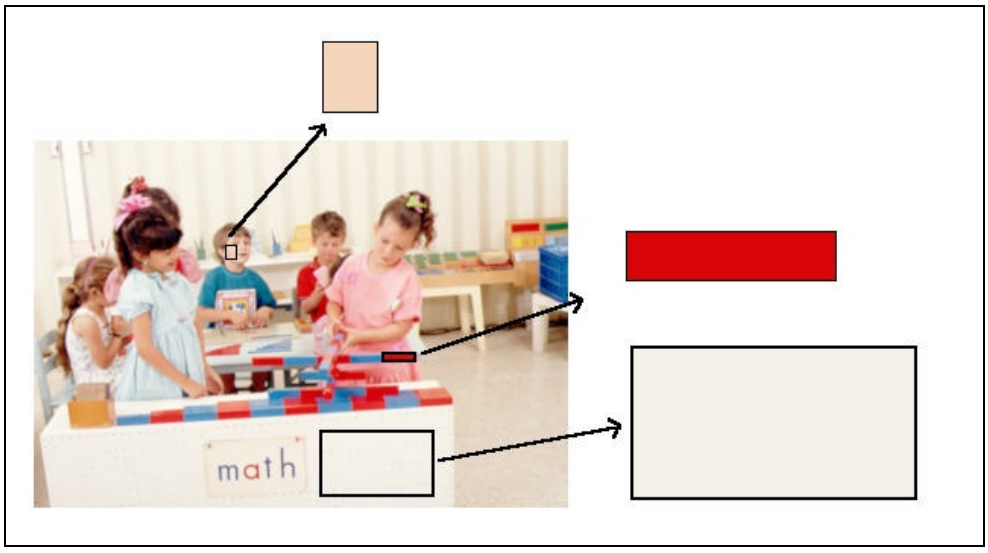
2. Multimedia information must usually be processed in real-time. For example, it is usually not possible to store a digital video that lasts one hour and process it during full day. Although a small delay in the processed signal is often tolerable, the algorithms used must be able to deal with the information at the same rate as it is input, so that the delay remains constant.

3. All the multimedia streams must be processed "in-sync", at the same time. There must be algorithms to deal with each of the media types, and all of them must share the available resources in order to transform the information in a synchronised way.

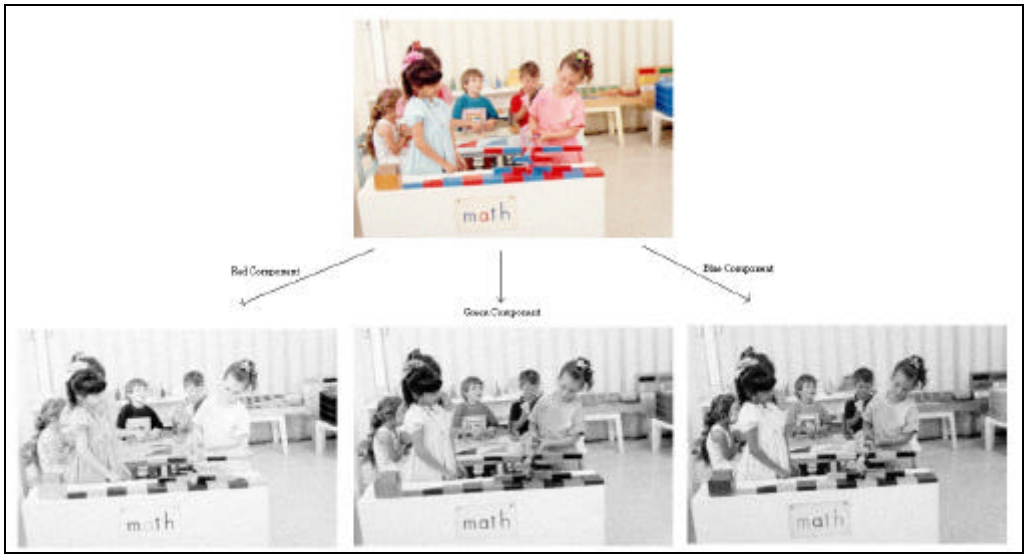
4. When there is a single entity producing multimedia information (such as, for example, when a film is broadcast over a network) a small delay is introduced in the multimedia stream in order to compensate for network jitters; this results in a much better perceived quality. If there is a two-way communication (such as in teleconferencing) the delays can be very annoying to the users, and the buffering technique cannot be employed.

5. Some special characteristics of video and sound media types make them amenable to compression:

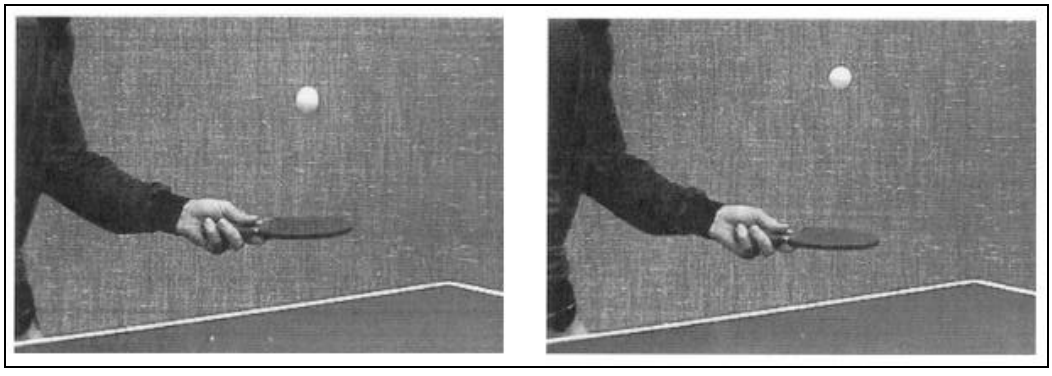
    - *Spatial correlation:* Within a single image or video frame, there is significant correlation among neighbouring samples (e.g. in an image, one pixel is usually stongly correlated to its nearest left neighbour) (see figure).



- *Spectral correlation.* For data acquired from multiple sensors (for example, in satellite images, or when considering the different colour components) there exists significant correlation among the samples from different sensors (see figure).



- *Temporal correlation.* For temporal data, such as video, there exist significant correlation among samples grabbed in near time segments (e.g. in video, the differences between consecutive images are usually very small, see figure).



The Need for Compression

Let us consider a video-based CD-ROM application. Full motion video at 30 frames per second and a 720 x 480 resolution generates data at 20.7 MByte/s. At this rate, only 31 seconds of video can be stored on a standard 650 MByte CD-ROM. With video compression techniques, up to 74 minutes of VHS-grade video quality can be stored in the same CD-ROM. With sound information competing for storage space, the need is even greater.

3.1 DATA TYPES

The following table presents different types of media information and some use scenarios:

Media Type	Sample Format	Use Scenario
Single Image	JPEG, GIF, BMP, PCX	Pictures in web pages, pictures obtained with digital cameras and digitizers.
Video Stream	MPEG, AVI, MOV, QuickTime, RealVideo, Netshow	Video captured with frame grabbers, Internet video-clips, digital television.
Audio Streams	AU, RealAudio, AVI, MIDI, QuickTime	Internet radio, audio-clips, digital television.
Text information	AVI, H.323	Internet chat, Film subtitles.
3D Animations	VRML	3d Worlds

3.1.1 Still Images

There has been an explosion in the use of images in computers; on a hand, desktop computers communicate information primarily via their screens, and there have been enormous advances in the area of graphical-user interfaces. On the other hand, the increase in computer-processing performance has enabled the proliferation of image-related computer programs, such as desktop publishing, image enhancement, computer graphic design, tele-conferencing, remote surveillance, etc.

A computer represents a digital image as a rectangular array of pixels, each with a certain *luminance* and *chrominance* values. The luminance is a value indicating the pixel's intensity (naively, to indicate the brightness of the pixel; a low intensity value will indicate a dark pixel, whereas high intensity denotes a bright pixel). The chrominance measures the colour of the pixel, and there are several ways to express it, such as the amounts of red & green present in the pixel (the amount of blue can be determined from the luminance and the other two chrominance values). Black & white images are usually represented only with luminance values.

Two important parameters of a digital image are the image *resolution* and *bit depth*:

- Resolution
  This parameter measures the height and width in pixels of the images presented on screen. It is usually expressed as *width* x *height* pixels, although it would be more appropriate to use pixels^2. The range of values that this parameter can take is very wide; as extreme examples, we will mention the resolution of small iconic images used in web pages, which is about 25 x 25 pixels^2, and that of images captured with digital cameras, about 1024 x 1024 pixels^2.

  The following figure shows a real-world scene as represented with images with different spatial resolutions:

- Bit Depth
  This parameter indicates the amount of information that we store for each pixel in the image. It is usually measured in bits per pixel (bpp). For example, if we are dealing with a black & white image and we store, for each pixel in the image, one byte of information (that would enable us to represent 256 different gray levels for each pixel), the bit-depth would be of 8 bits per pixel.

  A very typical bit depth value is 24 bpp, with three bytes of information for each pixel in the image: one byte to indicate the amount of red, one for the amount of green, and other for the amount of blue (these three quantities together contain both the luminance and chrominance information, in what is called RGB representation of images).

  The following figure displays a real-world scene as represented by images with different bit depths.

3.1.2 Video Sequences

The most important parameters of digital video are:

- Frame Rate
  Frame rate is measured in frames per second (fps), and indicates the number of images (frames) that are rendered to the user screen each second. The illusion of motion can be experienced at frame rates of about 12 fps, but modern cinema uses 24 fps, and television 25 fps.

- Resolution
  This parameter has the same meaning as for single images, and measures the height and width in pixels of the images presented on screen. In digital video documents all the frames in a sequence have the same resolution. Digital video comparable to television requires dimensions of about 640 x 480 pixels2 (width of 640 pixels and height of 480 pixels).

- Bit Depth
  This parameter has the same meaning as for single images, and indicates the number of bits used to represent each pixel in the image. In digital video documents all the frames in a sequence have the same bit depth.

  The following table illustrates possible values of these parameters for typical applications of digital video.

Application	Frame Rate (fps)	Resolution (pixels^2)	Bit Depth (bpp)
Multimedia	15	320 x 240	24
Entertainment	25	640 x 480	16-24
Surveillance	5	640 x 480	8-24
Video-conference	10	320 x 240	8-12
HDTV	25	1920 x 1080	24

When the video is compressed one must also consider the artefacts introduced by the compression algorithm. To measure them there are several established quantitative measures; the most important are *absolute error per pixel* (average error between each pixel of the original image and the decoded image), *mean squared-error per pixel*, and *signal-to-noise ratio*. The first two take greater values for worse image qualities, and the last one takes greater values for better image qualities. Unfortunately these quantitative measures do not always reflect the image qualities experienced by human observers, and are seldom used in the practice.

3.1.3 Audio

The sound and music that a computer can play can be classified into two types: *synthesised (instrument) sound* or *wave sound*. Synthesised sounds are those produced through a synthesiser called synthesiser commands. In order to play a synthesised sound, the computer sends a command to the sound device indicating the instrument, the note and the duration, and the sound device plays it. A wave sound is a digital representation of an analog sound signal. When the sound is digitally recorded or created, samples of the wave are captured at fixed intervals; the higher the sample rate, the better the final quality of the recorded or produced audio.

Audio data is characterised by three parameters:

- Sampling rate
  Indicates the number of sound samples (a sample is a single value obtained by the analog-to-digital converter, which indicates the intensity of the sound at that precise moment in time) that are in a unit of time. It is generally expressed in Hertz (1 Hz =

1sample per second) or units derived from it, such as thousands of Hertz (KHz, kilohertz) or thousands of KHz (MHz, megahertz).
Sampling rates are always measured per channel; for example, a stereo data recording at 8000 Hz will actually obtain 16000 samples per second (16000 Hz, or 16 KHz).

Typical sampling rates are 8000 Hz (telephony standard) or 44100 Hz (CD sampling rate).

- Bits per sample
  This parameter indicates how many bits are used to store each audio sample. It is generally expressed in bits per sample, and typical values are 8, 12 or 16 bits per sample.

- Number of channels
  This parameter indicates the number of independent streams of audio information. Several channels are frequently used in stereo audio clips, or in multi-language documents.

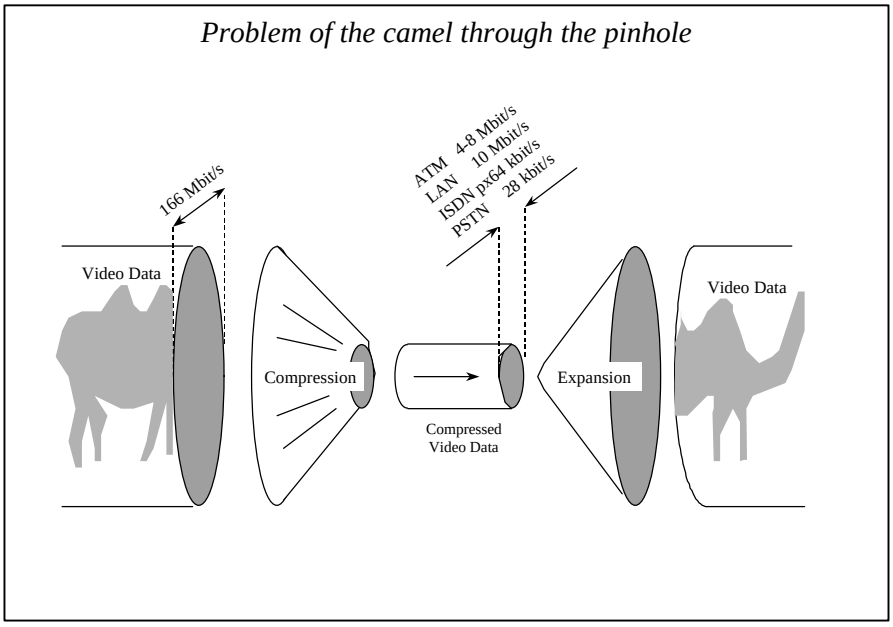
  The most frequently used numbers of channels are 1 channel for mono, and 2 channels for stereo.

3.2 COMPRESSION

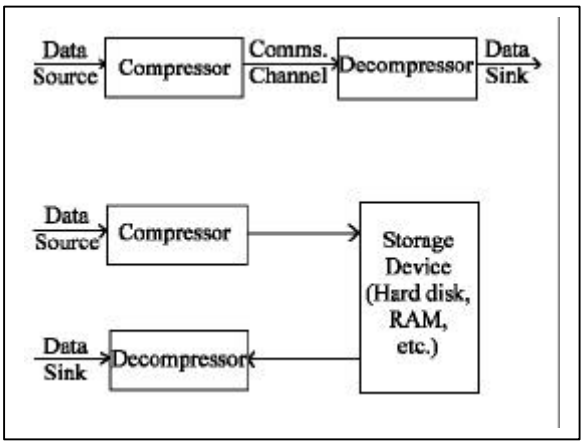
The viability of digital multimedia relies on image compression technology. The high data rates that result from the various types of digital multimedia content make their transmission and storage very difficult. Event entertainment video with moderate frame rates and image resolutions would require bandwidth far in excess of those available in single-speed CD-ROM. Similarly the data transfer required by a video-conference system is far superior to the bandwidth available over plain PSTN. Even with high-bandwidth technology, the per-byte cost of transmission would have to be very low before it could be used for the needs of HDTV.

The problem of multimedia data transmission is not unlike that of the camel through the pinhole:

Problem of the camel through the pinhole



There are also the problems of storage space and data processing of high volumes of digital information. To overcome these problems information theory has been used for long time in order to compress the information without significantly distorting it. The compression rates achieved are incredibly high, and there are continuous innovations in this field.



There are many compression techniques, but all of them fall into one of two categories: *lossless compression* and *lossy compression*:

- **Lossless compression**
  In lossless compression, the data obtained by the decompressor is identical to that introduced to the compressor; no bits of the compressed documents are changed. This kind of compression is used for binary files (e.g. executable programs, word processing documents), where corruption of a single bit could result in catastrophic consequences.

  Lossless coding techniques are generally restricted to compression factors of approximately 50% (i.e., the compressed files are usually about a 50% of the original ones); of course, the compression ratio depends of the redundancy present in the documents, and some of them compress better than others (for example, databases generally compress to less than 10%).

- **Lossy compression**
  In lossy compression, data obtained from the decompressor is not identical to that input to the compressor. Lossy compression techniques are not appropriate for binary files, but they can be used very effectively for multimedia information, where they attain much better compression ratios than lossless schemes.

  Successful lossy compressors are those in which the differences between the original documents and the decompressed ones are imperceptible by human observers. Thus development of lossy techniques is based on an understanding of the human visual and acoustic perception systems.

  Of course, different data types have different storage requirements. To appreciate the state-of-the-art in multimedia compression, the following table provides data rates, for different applications, of several media types, both uncompressed and compressed:

Media Type (Application)	Data Rate	
	Uncompressed	Compressed
Sound (Voice), 8000 samples per second, 8 bits per sample	64 kbps	2-4 kbps
Slow-motion video (Security surveillance),10 frames per second, 176x120 pixels, 24 bits per pixel	5,07 Mbps	8-16 kbps
Audio (Audio Conference), 8000 samples per second, 8 bits per sample	64 kbps	16-64 kbps
Video (Video Conference), 15 frames per second, 352x240 pixels, 24 bits per pixel	30,41 Mbps	64-768 kbps
Audio (Stereo Digital Audio), 44.100 samples per second, 16 bits per sample.	1.5 Mbps	128-1,5 Mbps
Video (Video File Transfer), 15 frames per second, 352x240 pixels, 24 bits per pixel.	30,41 Mbps	384 kbps
Video (Digital CD-ROM), 30 frames per second, 352x240 pixels, 24 bits per pixel.	60,83 Mbps	1,5-4 Mbps
Video (Broadcast), 30 frames per second, 720x480 pixels, 24 bits per pixel.	248,83 Mbps	3-8 Mbps
Video (High Definition Television), 30 frames per second, 1280x720 pixels, 24 bits per pixel.	1.33 Gbps	20 Mbps

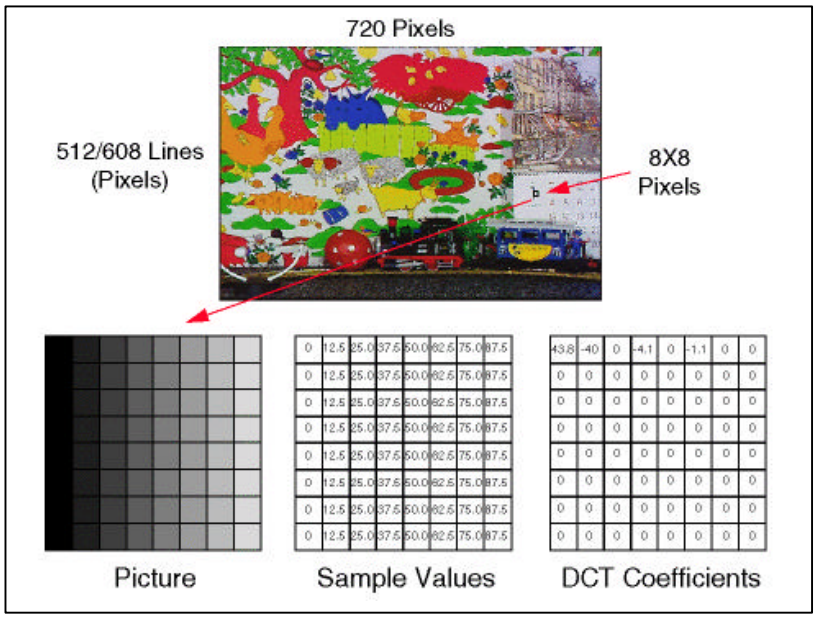
3.2.1 Still Image Compression

Image compression is basically a lossy process by which the information content of an image is reduced by eliminating redundancy in video signals. Compression techniques generally attempt to identify this redundancy and reduce a significant amount of it from the bit-stream. This manipulation of the image must be imperceptible for a human observer.

3.2.1.1 Identification of Redundancy with DCTs

The first step in most effective image compression systems is to identify the spatial redundancy present in the image. This is done by applying the Discrete Cosine Transform (DCT) throughout the image in blocks of pixels. The DCT is a lossless, reversible, mathematical operation which converts spatial amplitude data into spatial frequency data. This calculation is made on 8 by 8 blocks of luminance samples and the corresponding blocks of the colour components (see figure below). The DCT coefficient at the upper left is the represents the average value (of that

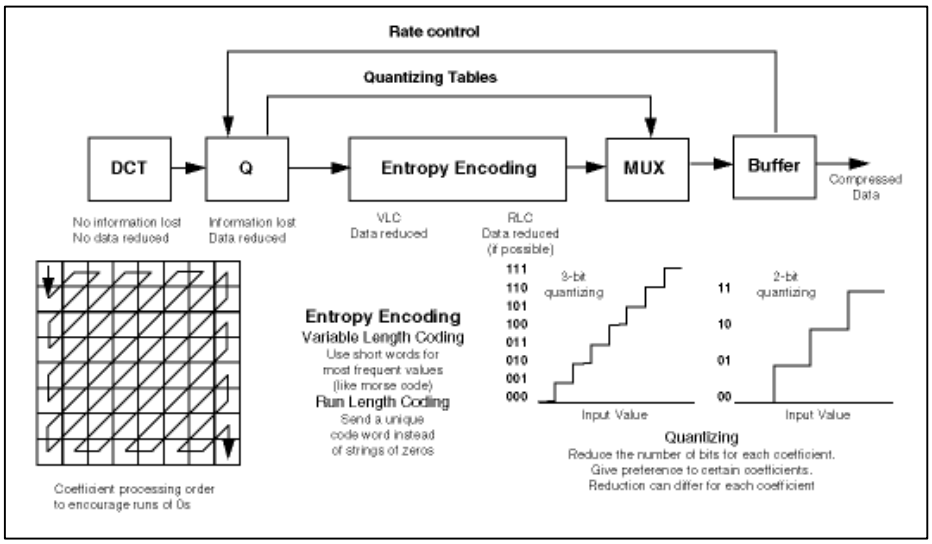
component) for the block. The other coefficients represent increasingly higher spatial frequencies.



It is the nature of images that a DCT-based transformation often results in very small values for many higher spatial frequency coefficients. Similarly, it is the nature of human visual perception that many of the non-zero, higher frequency, spatial coefficients can be quantized (i.e., represented by fewer bits) or completely dropped without noticeable degradation. It is important to notice that the DCT itself does not reduce the image data.

3.2.1.2 Intra-frame Compression

The actual compression begins with a reduction of the spatial redundancy. This is accomplished using intra-frame (inside the frame) compression (see figure). Intra-frame compression uses a combination of lossy and lossless processing to reduce the amount of information needed to represent one picture. We will describe each of these processes in more detail.



3.2.1.2.1 Quantisation

The power of image compression techniques comes from the clever quantisation of the DCT coefficients. Quantising is simply the process of reducing the number of bits which represent each coefficient. This is accomplished via a so-called quantisation table that contains, for each of the 64 DCT coefficient, a value by which this coefficient will be divided. This table will be stored in the compressed image, so that the decompressor will be able to multiply the quantised values by the quantisation table coefficients. In general, it takes much less data to send the tables and the heavily quantized coefficients than it would to send the original DCT coefficients.

3.2.1.2.2 Lossless Compression

Following quantization, lossless data reduction is applied using variable length coding (VLC) and run length coding (RLC). The order in which the coefficients are sent optimises the efficiency of this encoding process. Processing the 64 coefficients in the 8x8 blocks using a zigzag pattern maximises runs of zero values for more efficient compression.

- **Variable length coding**
  This process identifies common patterns (or words) in the data and uses fewer bits to code frequently occurring values and more words to code less frequently occurring values. Morse code is a form of VLC using very short sequences for frequently occurring letters such as "e" (one dot). Another example of VLC is the popular PC program PKZIP which uses the Lempel-Ziv-Welch (LZW) algorithm to compress data files. Like quantisation, VLC coding produces tables to map patterns to codes. These tables, combined with the mapped codes, generally take much less data than the original data patterns.

- **Run-length coding**
  This is a process by which a unique code word represents a repeating pattern such as zeroes. For example, a string of 25 zero values can be represented by the ESC character followed by the value 25 (the count) followed by the value zero. Thus, 25 bytes are compressed to only 3 bytes. Note that VLC and RLC are lossless encoding processes.

3.2.2 Video Compression

This section presents some of the techniques that are used in industry and international standards to compress video information.

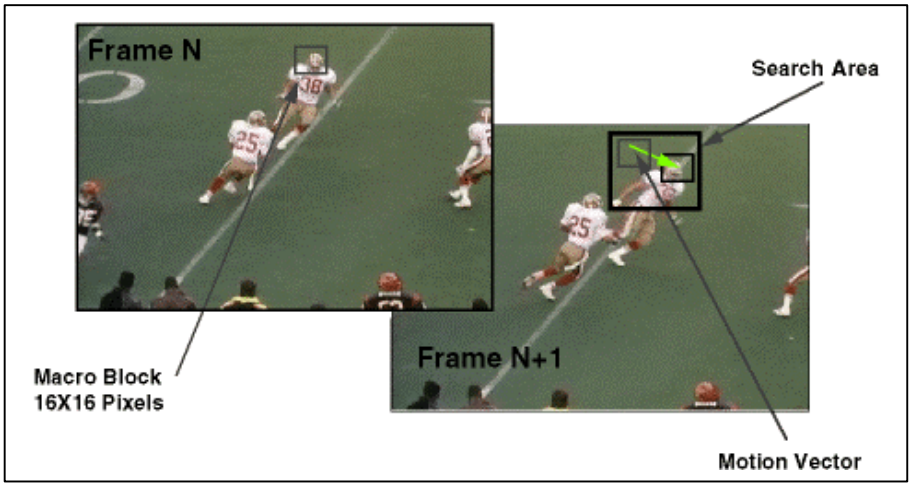
Since video is a sequence of images, video compression techniques are generally very similar to those used for images in order to eliminate spatial redundancy..

Another property of video signals is temporal redundancy which means that, for a given image sequence, the picture content generally varies little from frame to frame. The calculation of the relative picture content position changes (motion) between frames is an important part of inter-frame (between frame) compression (see figure).

3.2.2.1 Inter-Frame Compression

With intra-frame compression, a frame is compressed by compute its differences with the previous one, and eliminating the parts of the image that have not changed.

Also future frames can be used to compress the current frame; this is explained in the MPEG compression section [Section 3.3.2.1].



3.2.2.2 Motion Compensation

Consider the case of a video sequence where nothing is moving in the scene. Each frame of the video should be exactly the same as the previous one. In a digital system, it should be clear that, we only need to send one frame and a repetition count. Consider now, a dog walking across the same scene. The scene is the same throughout the sequence, but only the dog moves. If we could find a way of only sending the motion of the dog, then we can save a lot of storage space. This is an oversimplified case of motion video, but it reveals two of the most difficult problems in motion compensation :

- How can we tell if an image is stationery ?
- How do we extract the part of the image that moves ?

We can try to answer these questions by some form of comparison between adjacent frames of the sequence. We can assume that the current and previous frames are available for the comparison. The simple comparison technique is too simple and is like a frame-by-frame DPCM. This has a few problems :

- The pixel compare will rarely produce a zero difference, due to quantisation noise in the system (this can be overcome with a threshold).
- Images are rarely stationery.

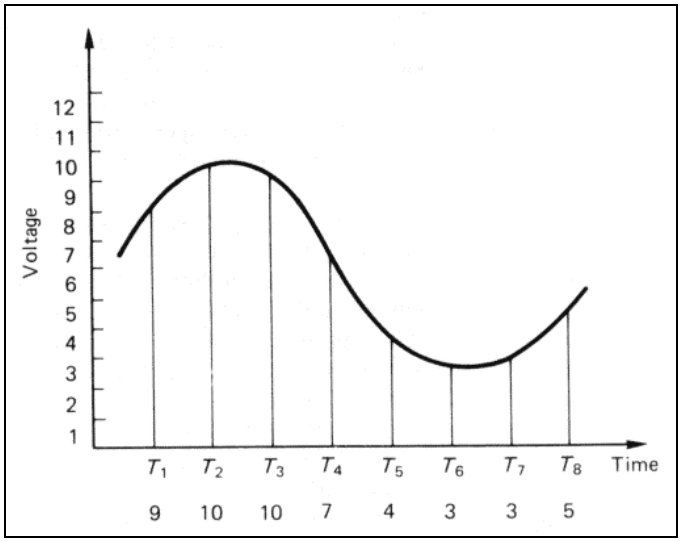
Therefore, more sophisticated techniques are needed. This problem is usually addressed by dividing the image into blocks. Each block is examined for motion. If a block is found to contain no motion, a code is sent to the decompressor to leave the block the same as the previous one.

If enough processing power is available, still more powerful techniques may be applied. For examples, blocks may be compared to previous block to see if there is a difference between the two. Only this difference (*motion vector*) is sent.

The motion estimation process in video compression consists of dividing the picture into macroblocks which are 16 by 16 pixels (four 8x8 blocks) and a search carried out to determine its location in a subsequent frame. Although the samples in the macroblock may have changed somewhat, correlation techniques are used to determine the best location match down to one-half pixel. A successful search will result in a motion vector for that macroblock.

3.2.3 Audio

Although there are several different ways in which audio (and video) signals can be represented digitally, there is one system, called Pulse Code Modulation (PCM), which is in used virtually everywhere. The following figure shows how PCM works. A series of discrete, equally spaced steps are marked on the time axis; these steps indicate the moments in time when the audio waveform is sampled. The voltage (represented by the height in the figure) of each sample is then described by a whole number that must fit in a number of bits indicated by the sample precision.



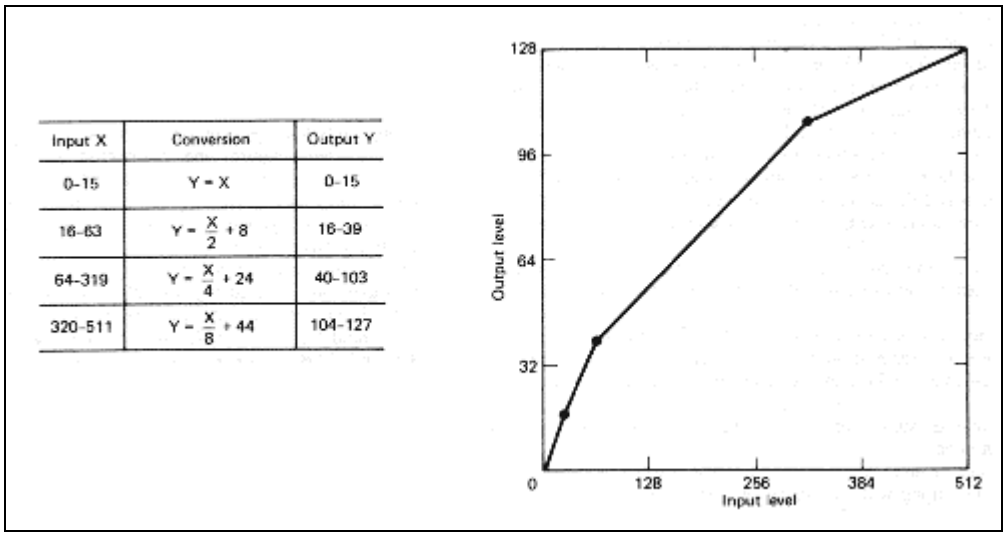
Let us examine some techniques that are used to compress audio information.

- Silence compression
  Silence compression on audio files is the equivalent to run-length encoding of general raw-data files. The compressor takes the input sound and eliminates those parts with silence (or with background noise), replacing them with an indication of the silence duration.

  This compression technique is “lossy”, since it replaces sequences of relative silence with sequences of absolute silence.

- Non-uniform coding
  This technique is used is 8mm video films. Basically, the system reduces the number of bits used to represent each sample from 10 to 8 bits. In the 10-bit input audio signal there are 1024 possible values, symmetrical about the centre of the range: half of them (512) are positive, ant the other half is negative. The 10 bits samples are expressed in signed binary, using 1 bit to express the sign and the remaining 9 bits to express the magnitude. The 9-bit magnitude is compressed to 7 bits, to which the sign bit is added to obtain the 8-bit encoding.

  The encoding of the 9-bit magnitude in 7 bits is performed as follows:



Input X	Conversion	Output Y
0–15	$Y = X$	0–15
16–63	$Y = \frac{X}{2} + 8$	16–39
64–319	$Y = \frac{X}{4} + 24$	40–103
320–511	$Y = \frac{X}{8} + 44$	104–127

The first 16 input levels (0-15) are left unchanged; the next 48 input levels (values 16-63) are given a gain of one-half, producing 24 different output levels taking from 16 to 39. The next 256 levels (64-319) are assigned a gain of one-quarter, thus giving 64 different output levels from 40 to 103. The remaining 208 levels (320-511) are encoded with a gain of one eighth, to produce outputs from 104 to 127. This compression results in an progressive loss of precision as the amplitude of the wave increases.

Of course, this process is reversed on reproduction of the audio signal.

3.3 STANDARDS

A lot of standardisation activity has been carried on, and is still under development in order to achieve effective and interoperable multimedia information exchange means. The current state of the art is presented in this section.

3.3.1 Digital Video Studio

Broadcast TV studios have dealt with digital video since long ago. CCIR (International Consultative Committee for Radio) Recommendation 601 defines a digital video format for TV studios for 525-line and 625-line systems. With the aid of this standard, the international exchange of production-quality programs is facilitated. It is based on component video with one luminance and two colour chrominances. The parameters for CCIR-601 are tabulated below; note that the raw (uncompressed) data rate for the CCIR-601 formats is 165 Mbps. Because this rate is excessive for most applications, the CCITT (International Consultative Committee for Telephone and Telegraph) Specialist Group XV has proposed a new digital video format, CIF (Common Intermediate Format), whose parameters are also shown below. Finally, the MPEG video standard defines another format.

	CCIR-601 625/50 PAL-SECAM	**CCIR601 525/60 NTSC**	**CIF**
Image width			
• Luminance	720	720	360
• Chrominance	360	360	180
Image height			
• Luminance	576	480	288
• Chrominance	576	480	144
Frames per second	50	60	30

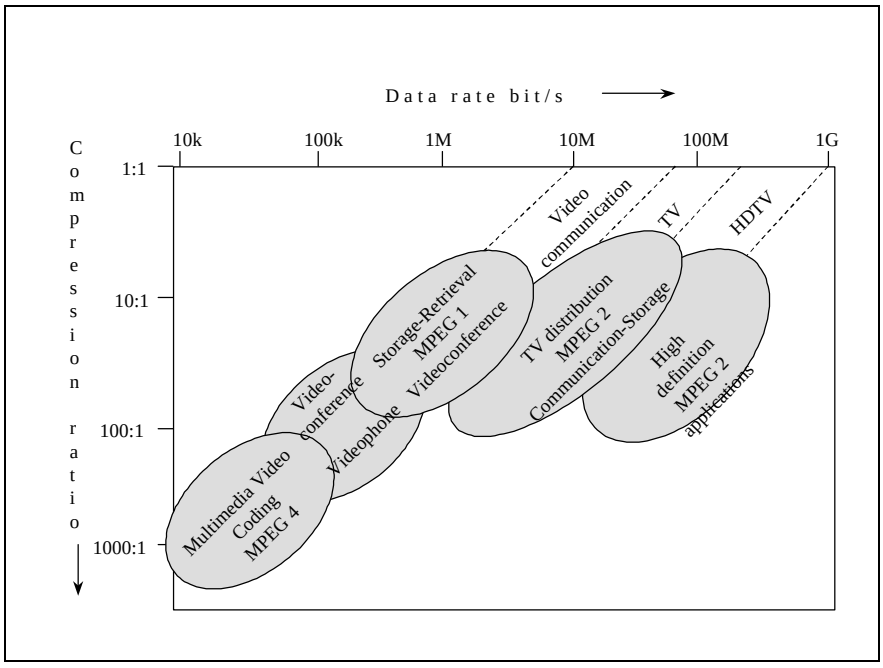
Other very common standard image resolutions are those derived from standards for computer display resolutions, set by VESA (Video Electronics Standards Association). The older personal computer standards are VGA (640 x 480 pixels^2) and TARGA (512 x 480 pixels^2). Today most workstations conform to the S-VGA standard (1024 x 768 pixels^2).

3.3.2 Video Compression Standards

The feasibility of digital video depends on how well it is compressed. The conclusion of researchers in digital video compression is that the quality of reconstructed CCIR-601 video after compression by a factor of more than 100 is comparable to VHS analog videotape quality. Standardisation of effective video compression techniques ensures compatibility of digital video equipment and applications. The most important video compression international standards are summarised in the following table together with their bandwidth requirements, and then described individually.

Standard	Bandwith
MPEG-1	1.5 Mbps
MPEG-2	10-20 Mbps
H.261	Multiples of 64 kbps [p x 64 kbps (p=1, … , 30)]
H.263	28.8 kbps (v.34 Modem)

The following diagram shows the roadmap of standard video compression formats:



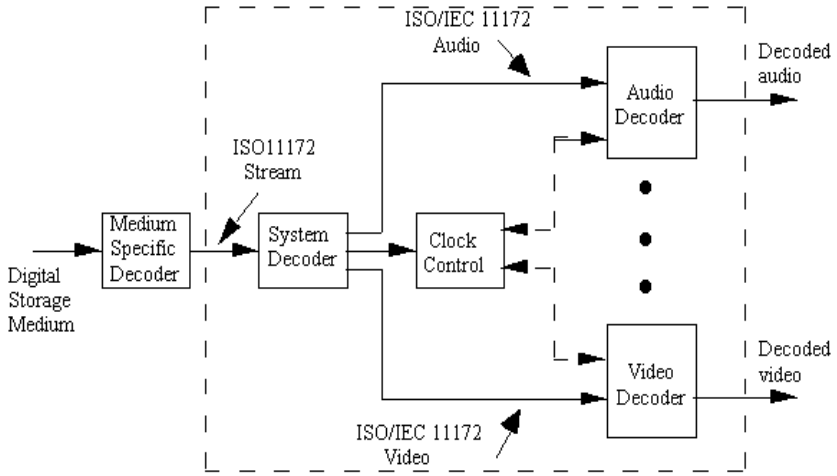
3.3.2.1 MPEG-1

Formally referred to as ISO-11172, MPEG-1 consists of five parts; we will present a brief overview of each of them, and then give an introduction to MPEG-1 compression and bitstream format:

3.3.2.1.1 11172-1: Systems

Part 1 addresses the problem of combining one or more data streams from the video and audio parts of the MPEG-1 standard with timing information to form a single stream, as depicted in the following figure. The combined data stream will be stored or transmitted as digital

document, and the decoder will have to split the stream to obtain the different streams, as shown in the following figure:



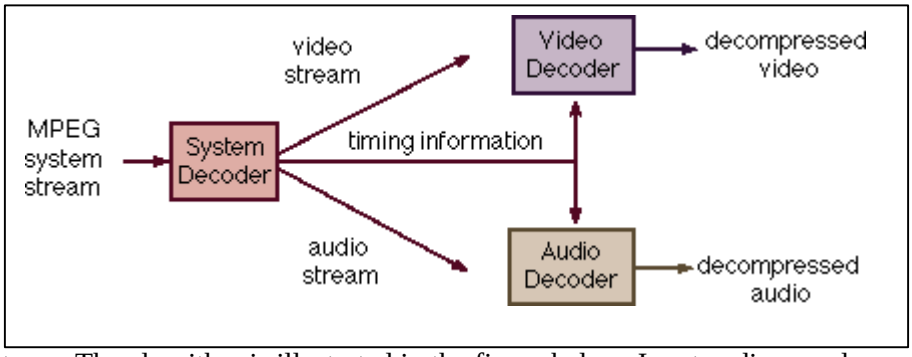
3.3.2.1.2 11172-2: Video

Part 2 specifies a coded representation that can be used to compress video sequences, in 625 and 525 line formats, to bitrates about 1,5 Mbits per second. This part was developed mainly to operate from storage media offering a continuous data rate of 1,5 Mbits per second, but the approach is generic and can be used more widely.

Different techniques are used to achieve a high compression ratio. The first one is to select an appropriate spatial resolution for the signal. The algorithm then uses a block-based motion compensation algorithm to reduce the temporal redundancy. Motion compensation is used to predict the current picture from a previous one, for prediction of the current picture from a future one, or for prediction of the current picture from both past and future pictures. The difference signal is further compressed using the discrete cosine transform to remove spatial correlation and is then quantised. As a last step, the motion vectors are combined with the DCT information, and coded with variable-length codes.

3.3.2.1.3 11172-3: Audio

Part 3 specifies a coded representation that can be used for compressing audio sequences - both



mono and stereo. The algorithm is illustrated in the figure below. Input audio samples are fed into the encoder. The mapping creates a filtered and subsampled representation of the input audio stream. A psychoacoustic model creates a set of data to control the quantiser and coding. The quantiser and coding block creates a set of coding symbols from the mapped input samples. The block 'frame packing' assembles the actual bitstream from the output data of the other blocks, and adds other information (e.g. error correction) if necessary.

3.3.2.1.4 11172-4: Conformance

Part 4 specifies how tests can be designed to verify whether bitstreams and decoders meet the requirements as specified in parts 1, 2 and 3 of the MPEG-1 standard. These tests can be used by:

- manufacturers of encoders, and their customers, to verify whether the encoder produces valid bitstreams.
- manufacturers of decoders and their customers to verify whether the decoder meets the requirements specified in parts 1,2 and 3 of the standard for the claimed decoder capabilities.
- applications to verify whether the characteristics of a given bitstream meet the application requirements, for example whether the size of the coded picture does not exceed the maximum value allowed for the application.

3.3.2.1.5 11172-5: Software

Part 5, technically not a standard, but a technical report, gives a full software implementation of the first three parts of the MPEG-1 standard. (The source code is not publicly available.)

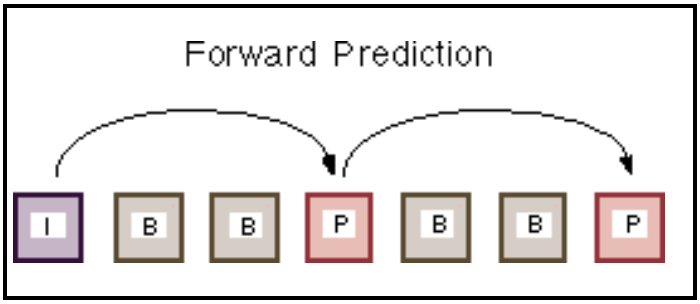
Initially, this standard was intended for video coding at rates of 1.2 Mbit/s, with stereo audio coding at bitrates of 250 kbits/s. We will concentrate on the video part. MPEG defines a resolution and frame rate called SIF (source input format, 352 x 240 pixels^2 at 30 fps, or 352 x 288 pixels^2 at 25 frames per second), which is not required: the format can accommodate much larger pictures and frame rates. MPEG includes functions to support fast forward, fast reverse, and random access within the compressed video document.

MPEG-1 basically specifies the video bitstream syntax and the corresponding decoding process, and includes methods to exploit both spatial and temporal correlation [see Section 2.2]. Spatial redundancies are exploited by using the discrete cosine transform (DCT) on image blocks that are 8 x 8 pixels^2, followed by quantisation, zigzag scan and variable-length coding of runs of zero coefficients, in a process very similar to JPEG still-image compression. Temporal redundancy is exploited by using motion compensated prediction.

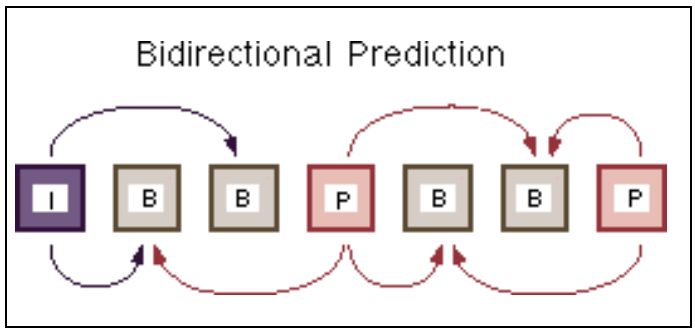
MPEG-1 supports three types of coded pictures:

- Intra (I) pictures
  These pictures are coded in a way similar to JPEG, so that all the information on the picture is present, without reference to other pictures in the video document.

- Predictive (P) pictures
  These pictures are coded with respect to the immediate previous I or P pictures. This kind of coding is called “forward prediction coding”.



- Bidirectionally Predictive (B) pictures
  These are coded with respect to both the immediate previous I or P picture, and immediate next I or P picture. This kind of coding is called “bidirectional predictive coding”.

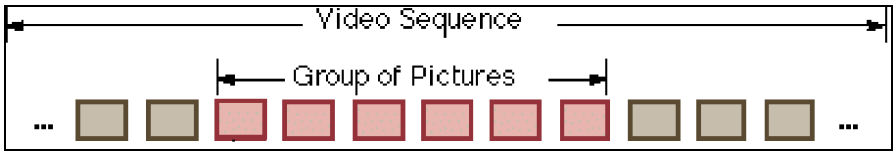


Obviously, in terms of compression efficiency I pictures are the least efficient, P pictures compress better (since they can be coded omitting the parts of the picture that have not changed since the previous frame), and B pictures the most efficiently coded (since they can be coded omitting the information that has not change since the previous frame, as well as the information that will be present in the next frame).

The following table summarises typical compression performance for the different types of pictures:

Type of Picture	Typical Compressed Size	Compression Ratio
Intra (I)	18 KBytes	7:1
Predictive (P)	6 KBytes	20:1
Bidirectionally Pred. (B)	2,5 KBytes	50:1
Average	4,8 KBytes	27:1

In typical MPEG-1 encoding, the input video sequence is divided into groups of pictures (GOP), where one GOP consists of an arrangement of one I-picture, several P-pictures, and several B-pictures.



3.3.2.2 MPEG-2

Formally known as ISO 13818, MPEG-2 is a standard currently in 9 parts. Most have reached International Standard status; part 10 is under development and scheduled to reach the IS status on February 1999. Part 8 was withdrawn.

3.3.2.2.1 13818-1: Systems

Part 1 addresses the problem of combining of one or more elementary streams of video and audio, as well as other data, into single or multiple streams that are suitable for storage or

transmission. This is specified in two forms: the Program Stream and the Transport Stream. Each is optimised for a different set of applications.

The Program Stream is similar to MPEG-1 Systems Multiplex. It results from combining one or more Packetised Elementary Streams (PES), which have a common time base, into a single stream. The Program Stream is designed for use in relatively error-free environments and is suitable for applications that may involve software processing. Program stream packets may be of variable and relatively great length.

The Transport Stream combines one or more Packetized Elementary Streams (PES) with one or more independent time bases into a single stream. Elementary streams sharing a common timebase form a program. The Transport Stream is designed for use in environments where errors are likely, such as storage or transmission in lossy or noisy media. Transport stream packets are 188 bytes long.

3.3.2.2.2 13818-2: Video

Part 2 builds on the powerful video compression capabilities of the MPEG-1 standard to offer a wide range of coding tools. These have been grouped in profiles to offer different functionalities.

3.3.2.2.3 13818-3: Audio

Part 3 is a backwards-compatible multichannel extension of the MPEG-1 Audio standard.

3.3.2.2.4 13818-4: Conformance testing

Part 4 corresponds to part 4 of MPEG 1.

3.3.2.2.5 13818-5: Simulation software

Part 5 corresponds to part 5 of MPEG 1.

3.3.2.2.6 13818-6: Extensions for DSM-CC (full software implementation)

Digital Storage Media Command and Control (DSM-CC) is the specification of a set of protocols which provides the control functions and operations specific to managing MPEG-1 and MPEG-2 bitstreams. These protocols may be used to support applications in both stand-alone and heterogeneous network environments. In the DSM-CC model, a stream is sourced by a Server and delivered to a Client. Both the Server and the Client are considered to be Users of the DSM-CC network. DSM-CC defines a logical entity called the Session and Resource Manager (SRM) which provides a (logically) centralized management of the DSM-CC Sessions and Resources.

3.3.2.2.7 13818-7: Advanced audio coding

Part 7 is the specification of a multichannel audio coding algorithm not constrained to be backwards-compatible with MPEG-1 Audio.

3.3.2.2.8 13818-8: (Withdrawn)

Part 8 was originally planned to be coding of video when input samples are 10 bits. Work on this part was discontinued when it became apparent that there was insufficient interest from industry for such a standard.

3.3.2.2.9 13818-9: Real-time interface

Part 9 is the specification of the Real-time Interface (RTI) to Transport Stream decoders that may be used for adaptation to all appropriate networks carrying Transport Streams.

3.3.2.2.10 13818-10: DSM-CC conformance testing

Part 10 specifies how tests can be designed to verify whether DSM-CC bitstreams and decoders meet the specified requirements.

3.3.2.3 H.261

H.261 is video coding standard published by the ITU (International Telecom Union) in 1990. It was designed for data rates which are multiples of 64Kbit/s, and is sometimes called p x 64Kbit/s (p is in the range 1-30). These data rates suit ISDN lines, for which this video codec was designed for. H261 is currently the most widely used international video compression standard for video-telephony on ISDN telephone lines. The standard describes the video coding and decoding methods for the moving picture component of an audio-visual service at the rates of p*64 kbps where p is in the range of 1 to 30. The standard is most suitable for applications using circuit switched networks as their transmission channels. This is understandable as ISDN with both basic and primary rate access was the communication channel considered within the framework of the standard. H.261 is usually used in conjunction with other control and framing standards such as H221, H230, H242, H230.

The source encoder operates only on non-interlaced pictures. Pictures are coded as luminance and two colour difference components (Y,Cb,Cr). The Cb and Cr matrices are quarter the size of the Y matrix H.261 supports two image resolutions, QCIF which is 144x176 pixels and, optionally, CIF which is 288x352 pixels.

The main elements in such an encoder are:

- **Prediction**
  H261 defines two types of coding. INTRA coding where blocks of 8x8 pixels each are encoded only with reference to themselves and are sent directly to the block transformation process. On the other hand INTER coded frames are encoded with respect to another reference frame.

  A prediction error is calculated between a 16x16 pixels region (macroblock) and the corresponding recovered macroblock in the previous frame. Prediction error of transmitted blocks are then sent to the block transformation process. The criteria for transmitting or not transmitting a block is not specified in the standard and is left as part of the coding control strategy.

  H261 supports motion compensation in the encoder as an option. In motion compensation an area in the previous (recovered) frame is searched to determine the best reference macroblock. Both the predication error and the motion vectors specifying the value and direction of displacement between the encoded macroblock and the chosen reference are sent. Neither the search area nor how to compute the motion vectors are subject to standardisation. However, both horizontal and vertical components of the vectors must have integer values in the range +15 to -15.

- **Block transformation**
  In block transformation, INTRA coded frames as well as prediction errors are composed into 8x8 blocks. Each block is processed by a two-dimensional FDCT (Forward Discrete Cosine Transform) function.

- **Quantisation**
  The purpose of this step is to achieve further compression by representing the DCT coefficients with no greater precision than is necessary to achieve the required quality. The number of quantizer are 1 for the INTRA dc coefficients and 31 for all others.

- **Entropy coding**

Here extra compression (lossless) is done by assigning shorter code-words to frequent events and longer code-words to less frequent events. Huffmann coding is usually used to implement this step.

- **Multiplexing**
  The video multiplexer structures the compressed data into a hierarchical bit stream that can be interpreted universally. The hierarchy has four layers:

  1. **Picture layer:** corresponds to one video picture (frame)
  2. **Group of blocks:** corresponds to 1/12 of CIF pictures or 1/3 of QCIF
  3. **Macro blocks:** corresponds to 16x16 pixels of luminance and the two spatially corresponding 8x8 chrominance components.
  4. **Blocks:** corresponds to 8x8 pixels.

- **Error Correction Framing**
  An error correction framing structure is described in the H261 standard. The frame structure is shown in Figure 2.2. The BCH(511,493) parity is used to protect the bit stream transmitted over ISDN and is optional to the decoder. The fill bit indicator allows data padding thus ensuring the transmission on every valid clock cycle.

Though H261, as mentioned before, is arguably the most widely used video compression standard in the field of multimedia conferencing, it has its limitations as far as its suitability for transmission over PSDN: H261 does not map naturally onto hierarchical coding. A few suggestions have been made as to how to make this possible, but as a standard there is no support for such schemes. H261 resolution is fine for conferencing applications. For applications requiring a higher quality video data, the upper limit optional CIF resolution, can be quite inadequate.

3.3.2.4 H.263

H.263 was designed for low bit rate communication, early drafts specified data rates less than 64 Kbits/s, however this limitation has now been removed, and the standard is used for a wide range of bit rates, not just low bit rate applications. It is expected that H.263 will replace H.261 in many applications.

H.263 is one of the best methods available today, when it comes to video compression efficiency. The coding algorithm of H.263 is similar to that used by H.261, however with some improvements and changes to improve performance and error recovery:

- Half pixel precision is used for motion compensation whereas H.261 used full pixel precision and a loop filter.

- Some parts of the hierarchical structure of the data stream are now optional, so the codec can be configured for a lower data rate or better error recovery.

- There are now four optional negotiable options included to improve performance: Unrestricted Motion Vectors, Syntax-based arithmetic coding, Advance prediction, and forward and backward frame prediction similar to MPEG called P-B frames. When using the advanced negotiable options in H.263 we can often achieve the same quality as H.261 with less than half the number of bits.

- H.263 supports five resolutions. In addition to QCIF and CIF that were supported by H.261 there is SQCIF, 4CIF, and 16CIF. SQCIF is approximately half the resolution of QCIF. 4CIF and 16CIF are 4 and 16 times the resolution of CIF

respectively. The support of 4CIF and 16CIF means the codec could then compete with other higher bitrate video coding standards such as the MPEG standards.

The four negotiable options included to improve performance are described below:

1. **Unrestricted Motion Vector mode.**
In this mode motion vectors are allowed to point outside the picture. The edge pels are used as prediction for the "not existing" pels. With this mode a significant gain is achieved if there is movement along the edge of the pictures, especially for the smaller picture formats. Additionally, this mode includes an extension of the motion vector range so that larger motion vectors can be used. This is especially useful in case of camera movement.

2. **Advanced Prediction mode.**
This option means that overlapped block motion compensation is used for the P-frames. Four 8x8 vectors instead of one 16x16 vector are used for some of the macroblocks in the picture, and motion vectors are allowed to point outside the picture as in the UMV mode above. The encoder has to decide which type of vectors to use. Four vectors use more bits, but give better prediction. The use of this mode generally gives a considerable improvement, especially subjectively because OBMC results in less blocking artefacts.

3. **Syntax-based Arithmetic Coding mode.**
In this mode arithmetic coding is used instead of VLC coding. The SNR and reconstructed frames will be the same, but generally fewer bits will be produced. In Telenor's encoder, the average gain for inter frames is 3-4%. This gain depends on the sequence, the bit rate and other options used. For intra blocks and frames, the gain is higher, on average about 10%.

4. **PB-frames mode.**
A PB-frame consists of two pictures being coded as one unit. The name PB comes from the name of picture types in MPEG where there are P-pictures and B-pictures. Thus a PB-frame consists of one P-picture which is predicted from the last decoded P-picture and one B-picture which is predicted from both the last decoded P-picture and the P-picture currently being decoded. This last picture is called a B-picture, because parts of it may be bi-directionally predicted from the past and future P-pictures. For relatively simple sequences, the frame rate can be doubled with this mode without increasing the bit rate much. For sequences with a lot of motion, PB-frames do not work as well as B-pictures in MPEG. This is because there are no separate bi-directional vectors in H.263, the forward vectors for the P-picture is scaled and added to a small delta-vector. The advantage over MPEG is much less overhead for the B-picture part, which is really useful for the low bit rates and relatively simple sequences most often generated by videophones.

These options are *negotiable.* This means the decoder signals the encoder which of the options it has the capability to decode. If the encoder has any of these options, it can then turn them on, and for each of the options used the quality of the decoded video-sequence will increase.

3.3.3 Audio

3.3.3.1 Wave Audio Representation

The audio wave is represented by a discrete approximation to the continuous signal.

Wave form audio file format (WAV), also called RIFF WAVE format, is an industry standard for wave sound representation. Microsoft Corporation developed it together with its Windows 3.1 operating system extension in order to provide sound support.

3.3.3.2 Synthesised Audio (MIDI)

Music Instrument Digital Interface (MIDI) is the industry standard for transmitting musical information between electronic instruments (keyboards, drum machines, electric guitars, sound-enabled computers, etc.). MIDI is only able to produce synthesised music, so that one cannot record, for example, a human voice, in MIDI format. This format has the advantage that the representation of music information is very compact, resulting in very small documents. The quality of the resulting sound does not depend on the format itself, but on the playing device.

3.4 TOLERANT TIME-STAMPING: TRANSFORMATIONS AND INVARIANTS

3.4.1 Tolerant Time-Stamping Of Multimedia Objects

It is important to be able to verify the time-stamp of a digital document even after it has suffered some suitable transformations. There are at least two scenarios where this tolerance property is desirable:

- The multimedia information is presented to an audience, or broadcast over a channel, and in the process there can be a degradation or modification of the document.

- The time-stamp is embedded in the multimedia document using digital watermarks [Appendix B].

In order to be able to time-stamp multimedia objects in a tolerant way, we will need to define several abstract notions.

3.4.1.1 Multimedia Transformations

A multimedia transformation is a function that maps multimedia objects into multimedia objects of the same type.

Examples of multimedia transformations are image rescalings (multimedia type: images and video), image cropping (multimedia type: images and video), echoing (multimedia type: sound), etc. We will enumerate some important multimedia transformations later in this document.

3.4.1.2 Feature Extractor

We will call feature extractor to a function that maps multimedia objects to ordered sets of real numbers.

The idea behind this definition is that a feature extractor extracts the “essential features” from a multimedia object. If the multimedia object is modified in a “non-essential” way, the feature extraction will produce the same feature values.

An example of feature extractor would be the function that assigns to a given 640 x 480 image the first 25 highest-energy (top left part of the spectrum) DCT coefficients. This feature extractor would provide the same values for several images that the eye would consider “equivalent”, since the human visual system tends to mask the higher frequency components in an image.

Another example of feature extractor would be a complex voice recognition device that would encode a sound as UNICODE characters representing the spoken words. In this case, the voice and intonation of the sentences are considered “inessential”, and therefore eliminated by the feature extractor. Of course, this characteristics will be essential in certain usage scenarios: we are just stressing the point that one can filter a multimedia document and keep only the interesting features for a specific application.

There are two very important sources for feature extractors: normalisation and invariants:

- **Normalisation**
  A very useful family of feature extractors will be those that *normalise* the multimedia object in a certain way. It is important to note that in this case the feature extractor produces another image (which is allowed by the definition of feature extractors), encoded as a sequence of real numbers.

  In the case of images, a possible *normalisation* would convert any image to a 200 x 200 pixel image with minimum intensity value 0 and contrast 1 (this can be achieved with a rescaling, a brightness shift and a contrast modification).

  The usefulness of this normalisation is that the size, brightness and contrast of an image are considered inessential features, and two images whose only difference is a rescaling, or a brightness change, or both, result in the same normalised version.

- **Invariants**
  Given a multimedia transformation, we will say that a feature extractor is an invariant for it if the result of the feature extraction is the same applying it on a multimedia object or in its transformation.

  Given a JPEG compression algorithm that compresses images to about one tenth their original size, it is possible to define an invariant for it, so that the same value is obtained on an image and in its JPEG-compressed version.

  We will list some invariants for common multimedia transformations later in this document.

3.4.1.3 Essentially Equivalent Multimedia Objects

Now that the features that are considered essential in a multimedia object can be obtained, it is possible to decide whether two objects are "essentially the same." Since the feature extractor associates each multimedia object with a vector of real numbers, it is possible to correlate the vectors associated to two multimedia objects, and to consider both multimedia objects "essentially equivalent" if the correlation obtained is sufficiently high.

> *Given a feature extractor F, a threshold real number $a>0$ and two multimedia objects O_1 and O_2 are said to <u>be essentially equivalent</u> if $Correl(F(O_1),F(O_2)) > a$. This will be indicated with the notation $O_1 \approx O_2$.*

3.4.1.4 Tolerant time-stamping

Once a feature extractors selected, it is possible to time-stamp not the multimedia objects themselves, but the vectors obtained with the feature extraction function. Therefore, to time-stamp a multimedia object one would first apply the feature extraction, and time-stamp the resulting vector; the same applies to certificate verification and renewal.

> *Given a feature extractor, one will say that a multimedia time-stamping algorithm is <u>tolerant</u> if the time-stamps are produced on the equivalence classes determined by the feature extractor.*

In verification, one would need to know the feature extractor used in time-stamp generation, and would need a threshold value in order to decide whether the correlation between the features extracted is sufficiently high.

3.4.2 Typical Transformations And Invariants For Them

In this section we will enumerate some common multimedia transformations and will provide some invariants for them. We only present image / video transformations, since their audio equivalents have analogous interpretations and invariants. For further information on detailed definitions of the transformations, please refer to any basic text on the concrete field (images, video, sound).

Transformation	Invariants
Rotation	• Average value of all pixels in the image • Intensity histogram • Number of lines (circles, ellipses, rectangles…) in the image (this can be obtained by thresholding generalised Hough transforms of the image) • Hotelling transform (normalisation with respect to rotations)
Scaling	• Average value of all pixels in the image • Relative intensity histogram • Normalisation with respect to size • Normalised highest-energy DCT / Fourier / Walsh / Wavelet coefficients • Number of lines (circles…) in the image
Brightness change	• Intensity histogram translated to origin • DCT coefficients (except DC value, which accumulates the brightness change and is the single one affected by it) • Image normalisation with respect to pixel intensity • Number of lines (circles…) in the image
Contrast change	• Normalised intensity histogram • Normalised DCT transform • Image normalisation with respect to pixel intensity
Imperceptible watermarking	• Depends on watermarking scheme, but in general an invariant could be obtained by selecting the features not affected by the watermarking scheme, or by a quantisation of those features affected by it
Blurring	• Highest-energy DCT / Fourier coefficients • Average value of pixels
Edge enhancement	• Lower DCT / Fourier coefficients • Average value of pixels
JPEG Compression	• Normalisation with respect to JPEG compression (the normalised image would be the JPEG compressed version of the image, compressed with a suitable, high, quantisation table).
MPEG Compression (Video only)	• Truncation of predictive and bi-directionally predictive frames, so that only intra-frames are preserved.

Of course, the list is not exhaustive, but it provides a clear idea of what is involved in the design of invariants.

4 TIME-STAMPING OF MULTIMEDIA INFORMATION

This section aims to identify situations of use of time-stamping services on multimedia information. Relevant (different) situations are identified, characterised as relevant for multimedia time-stamping, and recommendations are issued about the technique to be used.

The very first issue to deal with is that multimedia information is typically structured as a data stream. A stream is a potentially very long (or infinite) sequence of bits that are sent from the producer of multimedia information to the receiver. The stream is usually sent at a rate that is negotiated between the sender and receiver; an alternative is to use a demand-response protocol in which the receiver requests additional amounts of multimedia information. The main difference between streams and digital documents (also called "messages") is that the receiver must consume the received data at approximately the input rate; otherwise very large amounts of information would have to be buffered. In most applications where streams are used, the receiver only buffers a small amount of multimedia information, just enough to present a few seconds of audio and video. After presenting that information to the user, it will be discarded.

Some examples of digital streams are digitised audio and video, data feeds (news feeds, stock market quotes, weather information, etc.), and internet applets (although of finite size, these are best modelled as digital streams).

Traditional cryptographic operations are designed to be used on complete digital documents, and clearly need modifications in order to be used on digital streams; otherwise the receiver would have to store the complete digital stream before, for example, verification of the digital signature. In streams of infinite length (such as a 24-hour television broadcaster on the internet), this is impossible to achieve. Even for finite but very large streams the traditional approach is not practical. [GennRoh97], using the work in [EvGoMi94], propose several solutions to perform cryptographic signatures on digital streams, which we will adapt to the time-stamping operation.

4.1 WHOLE DATA

The most basic situation occurs when the multimedia document is closed (complete), it is to be handled as a unit (monolithic), and there is plenty of time to process (no real time requirement). In this case, the protocols for raw data time-stamping are to be used, as described in [PKITS D4]. Basically, the whole information is hashed, time information is added, and the authority (TSA) digitally signs it. Verification of the time-stamp follows the classical procedure, and the original document is required to reconstruct the hash value. It is important to notice that not a single bit may be modified in the original document, since it would alter the hash value, and make verification impossible. No lossy transformation may be applied to the document, neither for storage nor for transmission, even if the transformation does not alter the perceived content. It is important as well to notice that if the information has components (e.g. streams), these cannot be segregated without losing the time-stamp.

4.2 SEGMENTATION OF MULTIMEDIA STREAMS

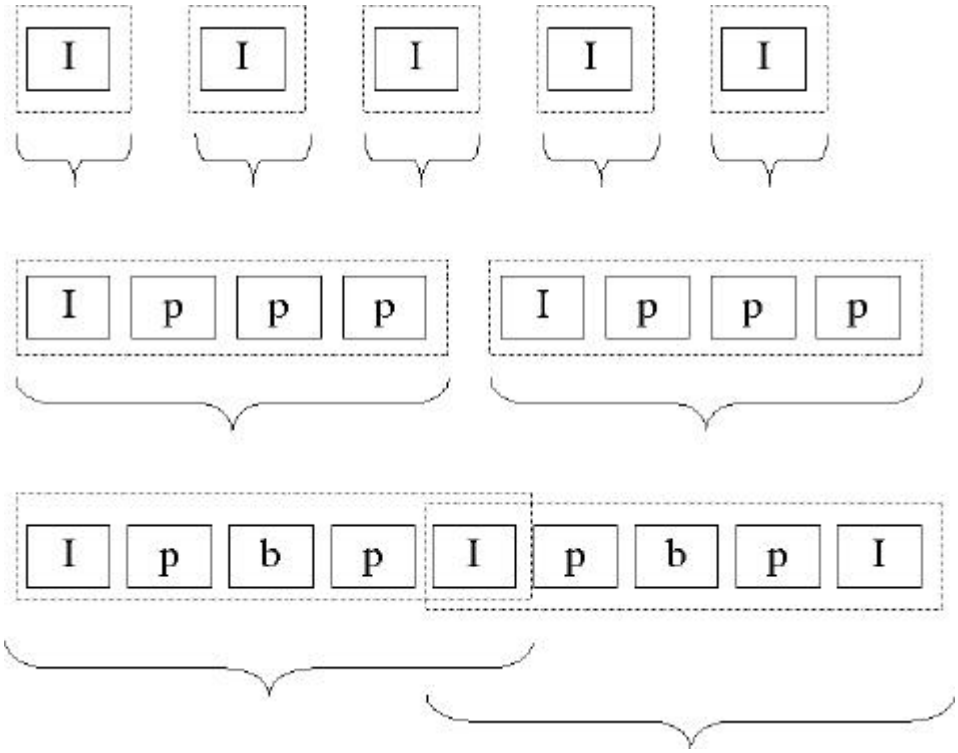
If the multimedia components may need separate treatment, the original document needs to be broken into parts before time-stamping. The multimedia information may be broken into smaller blocks, either along the time axis, or along the different streams of data, or both.

Segmentation may be needed as well when the whole stream is not known at the moment of time-stamping (e.g. in real time videoconference, or broadcasting of sport events, ...), and when the stream expands over a large period and finer time grain is needed: that is, each segment occurs at different instants in time.

This section presents criteria to segment streams, while the next sections will deal with actual time-stamping of individual segments, and time certificate storage and transportation.

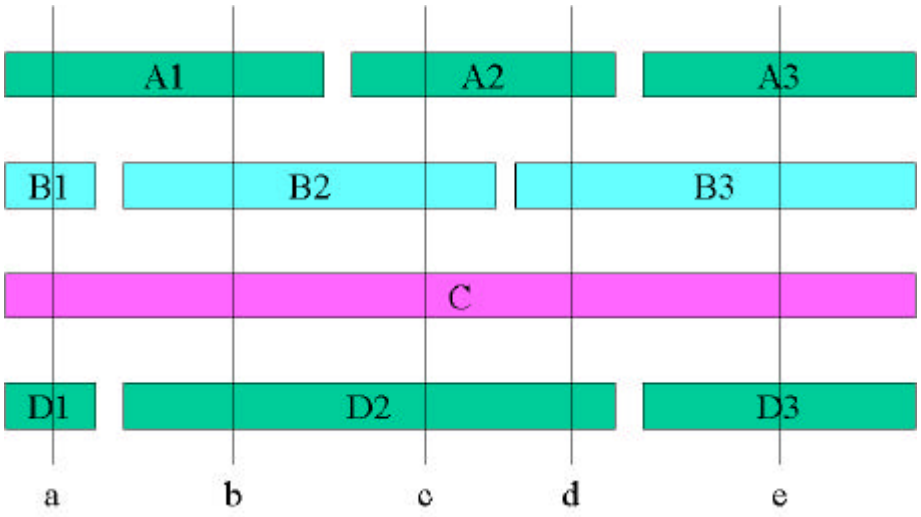
4.2.1 Temporal Stream Segmentation

Breaking along the time axis permits the reuse of components (clips) in isolation, while keeping streams synchronised (e.g. audio and video). Sensible breakpoints shall be selected: full frames for still images, and runs of intra (I) pictures in MPEG (or similar). There is no sensible criteria for audio information. In MPEG, P pictures are only meaningful with respect to the previous I picture, and its time-stamping shall be performed together. B pictures are harder since both previous and posterior information is needed; therefore, some overlapping is needed. The following picture describes units to time-stamp for a single MPEG stream.



When there are several streams, and synchronisation needs to be preserved, each stream imposes its own breaking points that may be the same ones or not. If different breakpoints are needed, each one shall impose a separate time-stamp. The picture below shows four data streams (A, B, C, and D), each one with its own meaningful breaking points (giving pieces A1, A2, A3, etc.). The procedure identifies breaking points for each stream, and collects relevant streams to synchronise at each point; in the example picture, the meaningful slices are labelled A, b, c, d and e. The following chunks of information are subject to time-stamping:

(a) {A1, B1, C, D1}
(b) {A1, B2, C, D2}
(c) {A2, B2, C, D2}
(d) {A2, B3, C, D2}
(e) {A3, B3, C, D3}



For audio streams, as any point is fine for breaking, breaking points shall be selected to match some adjoining MPEG stream, in order to reduce the number of time-stamping blocks.

Still picture streams may be broken either into single picture blocks (if atomicity is a requirement), or several pictures may be collected together into greater blocks determined by MPEG streams.

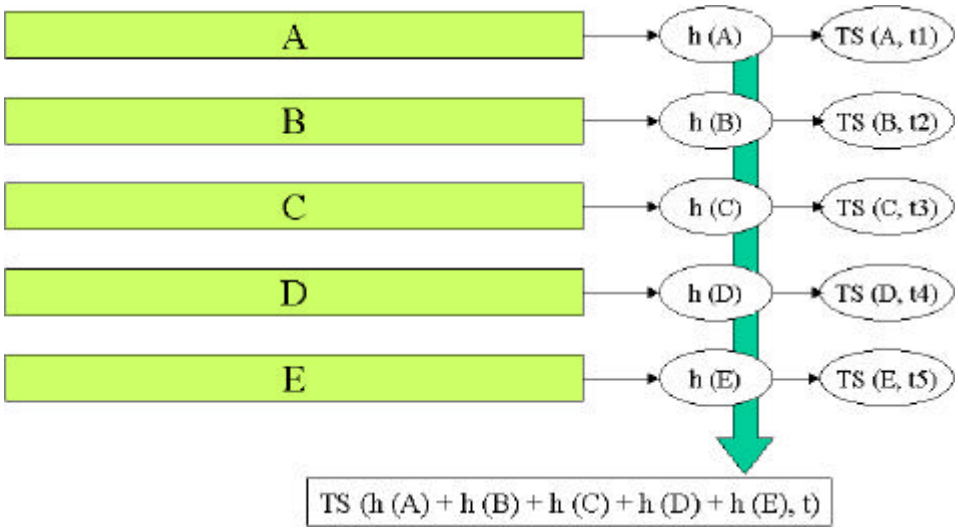
If the multimedia stream is to be broken apart into temporal segments, the block identifications are kept, and the subset of relevant time-stamp certificates has to be added.

This technique does not permit segregation of data streams, since lacking a single stream prevents time-stamp verification of the others.

4.2.2 Independent Streams

If different streams are intended to be used separately, or some stream may be replaced, or new streams may be incorporated in the future, the previous techniques do not help to keep streams independent.

In the simple case where temporal segmentation were not needed, the hash value of each stream is independently evaluated, and used to generate an individual time-stamp for that stream. Final bundling may be achieved by digitally time-stamping the individual hash values. If a stream is removed, its hash value must be retained for verification of the whole pack. If a new stream is added, it will be independently time-stamped, and a new overall time-stamp certificate must be generated for the new pack.



4.2.3 Bi-Dimensional Segmentation

This situation arises when both stream independence is needed, and streams may be temporally broken down. To address this scenario, first each stream is broken into segments as described above. Then, the hash value of each segment is evaluated, and time-stamped. Lastly, meaningful slices are identified, as described above, and the applicable hash values certificates at each instant are time-stamped together. Removal of a stream is feasible, as far as its hash value is retained for verification. Addition of a new stream requires a new series of hash values, their time-stamp certificates, and the generation of new time-stamps for temporal slices.

4.3 TIME-STAMPING OF SEGMENTED MULTIMEDIA STREAMS

After stream segmentation, a sequence of blocks is ready for time-stamping. Several alternatives may apply:

	pros	cons
time-stamping in isolation	▪ easy to use parts independently	▪ slow processing ▪ large data overhead
table of time-stamps	▪ blocks can be verified separately ▪ fast	▪ blocks cannot be separated ▪ table may be quite large
chaining with forward knowledge	▪ guarantees sequencing ▪ fast	▪ parts cannot be used separately ▪ the whole stream is required to start
chaining without forward knowledge	▪ guarantees sequencing ▪ only a short buffer is needed ▪ fast	▪ parts cannot be used separately ▪ bulky

4.3.1 Time-Stamping In Isolation

This solution works for both finite and infinite streams. The sender splits the stream in blocks of manageable length, and these blocks are time-stamped with conventional schemes (i.e., the stream blocks are time-stamped as traditional digital messages). A problem presented by this solution is that the sender must generate a signature for each block, and the receiver must verify one signature per block. With current digital signature algorithms at least one of these operations (signature or verification) will be computationally expensive (see Section x.x), thus

resulting in a bottleneck to the transmission rate of the stream. It could be argued that adding computing resources the transmission rate could be increased, but generally one wishes to devote all the available resources to the processing of the multimedia information, and we will attempt to find a more satisfactory solution.

4.3.2 Table Of Time-Stamps

This solution only works for finite streams. The stream is also split into blocks; instead of signing each block, the sender builds a table with the cryptographic hashes of every block, altogether with its time, and signs this table. When the receiver asks for the time-stamped digital stream, the sender begins by sending the table of block time-stamps, and the receiver can then check the time-stamps of the desired blocks.

A single digital signature verification has to be performed by the receiver for the whole digital stream, plus a hash computation per block.

A problem with this solution is that the table can be very large, and must be stored by the receiver during the transmission of the multimedia stream. For on-line verification, the whole table has to be sent in advance.

Example

Imagine that a 1-hour MPEG audio & video film at 30 frames per second is to be transmitted from sender to receiver, and that this second solution is used. If the blocks contain the information of 3 frames and their associated audio, we would have 10 blocks per second. If we also store the exact time associated to each block relative to the beginning of the film, we would need 2 bytes per block for the time. Using 20-byte hashes, the necessary information would be of 10 x (20 + 3) = 230 bytes per second. The whole table would occupy 230 x 60 x 60 = 828.000 bytes, almost one megabyte that the receiver would need to keep in memory during the whole transmission. (Of course, the scheme can be improved by using several tables for different parts of the stream, but we will present below more efficient mechanisms.)

4.3.3 Chaining With Forward Knowledge

In this scenario we assume that the multimedia information has finite size (possibly very large), and that it is completely known by the sender. This requirement is not so narrow as one could initially think, since it applies to most internet broadcasts (digital audio and video clips, for example). In [GennRoh97] it is proved that a single hash computation for each block will be sufficient; the general idea of their proposed solution is to embed in each block the accumulated time-stamp of the next blocks.

The multimedia stream is divided in blocks; the receiver will reserve a buffer of size sufficient to hold the received blocks. The sender transmits the time-stamp of the first block of multimedia information; after verification of the digital signature in the initial time-stamp, the receiver knows what the hash of the initial block should be. He then starts receiving the stream and computing the hashes of the received blocks. When the first block is received, its hash is compared to that in the received time-stamp; if they are different, then the transmission is aborted, otherwise processing continues. In order to authenticate and time-stamp the remaining

blocks, the hash and time of each block are embedded in the preceding one: the first block contains the hash and time of the second one, the second block contains the hash and time of the third, etc. With this scheme, only one digital signature must be checked for the whole digital stream, with an additional hash computation per block.

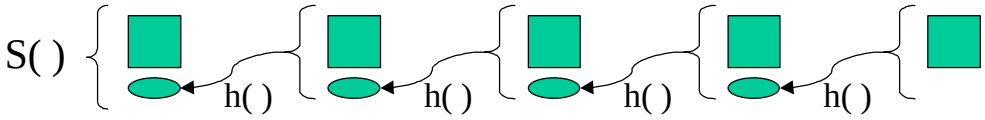
Formally, let $\{D_i, i= 1, ..., n\}$ be the sequence of data blocks to time-stamp, and H a hash function. The first blocks gets a full digital signature including its time reference, and the hash value of the next block:

$$S (D_1, T_1, H (D'_2))$$

From block 2 on, the information is chained:

$$D'_i= \{D_i, T_i, H (D'_{i+1})\}$$

One digital signature is enough to certify the whole stream. The receiver may verify blocks as they are received, without large buffering requirements.



4.3.4 Chaining Without Forward Knowledge

This alternative considers potentially infinite streams that are not known in advance to the sender; this includes streams such as videoconference audio and video, live television broadcasts, and internet chats.

It is based on one-time digital signatures: each block provides the parameters for signing the next block using a one-time digital signature protocol. In this case the solution proposed in [GennRoh97] is less optimal both in operations (requiring more operations to time-stamp a block) and size.

The stream is also divided into blocks. Initially, the sender sends a signed public key for a one-time signature scheme. He then sends the first block of multimedia information together with a one-time signature on its time-stamp based on the one-time public key. This first block also contains a one-time public key to be used the signature on the 2nd block. This structure is repeated for the remaining blocks, so that each one carries its one-time signature based on the one-time public key sent in the previous block, and a new one-time public key that will be used for the next block.

Formally, let $\{D_i, i= 1, ..., n\}$ be the sequence of data blocks to time-stamp, H a hash function, and $<K, S_K, V_K>$ a one-time signature scheme: K is the one-time signature parameters, S_K is the signing function, while V_K is the verifying function . The first blocks gets a full digital signature including its time reference, and the signing of the next block, including the parameters needed to verify the signature:

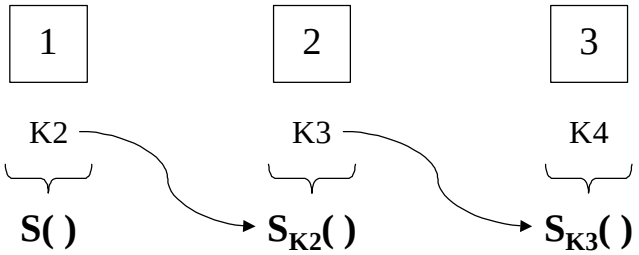
$$S (D_1, T_1, K_2)$$

From block 2 on, the information is chained:

$$S_{Ki} (D_i, T_i, K_{i+1}\}$$

The receiver may verify the first block using the public certificate of the signer. From block 2 on, the receiver may verify each block certificate using the signing parameters provided in the previous block.

One full digital signature is enough to certify the whole stream, but one-time signature verifications are also needed to be carried out for the blocks to be verified. The receiver may verify blocks as they are received, without large buffering requirements.



One-time signatures are expensive in terms of processing requirements, and introduce a significant amount of data that may cause a noticeable overhead. Still, one-time signature algorithms are faster than traditional RSA signing.

4.4 FAST TIME-STAMPING

Multimedia information has typically strong timing requirements that take as much processing power as available. Issuing time-stamps, and verifying them cannot take a significant amount of resources. Since time-stamping is basically a combination of evaluating hash values, and adding a digital signature, both cryptographic operations, must be efficient.

Hash functions are a compromise between evaluation time, hash size, and collision freedom. Fast evaluation is always a must. Hash size is again a compromise: it must be large to prevent collisions, but it cannot be a large amount to be added to the already bulky multimedia information. Lastly, collision freedom is subtle to analyse. If the information subject to hashing is strongly structured, it is much harder to find collisions than if any bits may be replaced. The issue with multimedia is that human perception allows for large modifications to go undetected. This means that were an effective collision attack devised, it would be hard to detect.

For raw-data time-stamping it was proposed to use more than one hash function (see [PKITS D4]). This may be an unacceptable overhead in multimedia scenarios.

For the time being, MD5 is the preferred function to hash multimedia information when processing power is limited. It is fast (see appendix 8.1), 16 bytes is a fair compromise, and no effective collision attacks are known.

Next to be preferred is SHA-1, that is noticeably slower (less than half bytes per second), but has a larger size, 20 bytes, and there are no known collision attacks. This algorithm is recommended when there is enough processing power available.

The case for signature algorithms needs to take into account the asymmetric role of signing and verifying.

4.4.1 Fast Verification

When there is plenty of time (processing power) to sign, but verification may be very fast, RSA is the winner algorithm. In this case, the hard key must be used for signing, and the cheap key for verification. A cheap key is one that has few bits on, so that the number of arithmetic operations is reduced. A value of 3 is recommended for the exponent of the public key. The

modulus is a compromise between security (large), and signature size (short). For commercial applications, 1024 bits is a reasonable compromise. 512 and 2048 bit sizes may be adequate for low value information, and for highly valued data, respectively.

In the future, elliptic curves may become a very interesting option both in terms of speed and in terms of security. However, research is still underway with respect to their robustness.

4.4.2 Fast Certification And Verification

Fast verification using public keys is achieved at the cost of slow signing. This is not acceptable when the signer is also time-stressed. The algorithms considered above (see section 5.3.3 and section 5.3.4) on chaining protocols for time-stamping present interesting properties regarding speed processing, that were already summarised.

The existence of only one full digital signatures reduces the overhead (in time and data) introduced by a heavy signing protocol. Light-weight chaining reduces the overhead for the rest of the stream.

4.5 TRANSFORMATION TOLERANT TIME-STAMPING

When the multimedia information may be subject to transformations that do not respect every bit, hash values cannot be used any longer. Instead a characteristic vector has to be used to replace the hash value in the signature schema. This introduces some problems:

- meaningful characteristics have to chosen
- verification depends both on the quality of the cryptographic primitives, and the interpretation of the correlation outcome
- more processing power is needed
- data volume grows up

These drawbacks are not usable with fast signing and verification scenarios: these are intrinsically bulky and slow.

Given a multimedia block to sign, D, and a set of descriptor functions to use F= <f1, f2, ..., fn>, a descriptor is created as F (D)= <f1 (D), f2 (D), ... fn (D)>. This descriptor is added in clear to the data, and then time-stamped: <D, S (F (D), T)>.

Signature verification is performed as usual on the descriptor data. To decide whether the time-stamp does really apply to the received multimedia information D', a new descriptor is created, F (D'), and correlated to the one provided by the signer. Correlation outcome, F (D) ≈ F (D') is inspected to decide whether to accept or reject data.

Usage of descriptors may add a non negligible amount of data to carry along the original multimedia information. A balance has to be found between very condensed descriptors (relatively easy to fake with colliding data), and very detailed descriptors (harder to subvert). When the descriptor is the multimedia information itself, we revert to the previous sections, where the multimedia information bits are strictly time-stamped.

4.6 TIME-STAMP CERTIFICATE TRANSPORT

Basically, a time certificate is a PKCS-7 object that is the digital signature of the data to be time-stamped. The formatting of this packet is to be performed according to [PKITS D4] conventions. Encoding of the PKCS-7 may be different for different multimedia packing standards. This section covers the issue of where are these packets stored and how are they carried along data for later verification.

Time-stamping services have a number of requirements that are briefly summarised below to focus the concerns of any transport mechanism that carries time certificates along with multimedia objects.

Time-stamping is a proof of existence, not of ownership. Any other party may later time-stamp the same information, and the same data may be time-stamped many times on different TSA by different requestors. Indirectly, time-stamps may be used to sort out ownership disputes, since the one showing the earliest time-stamp may claim that the other ones stole the data (or just abused the public availability of data to time-stamp them again).

Time-stamping is a proof of knowledge, useful for non-repudiation: the requestor cannot argue ignorance of something he ordered to time-stamp. Actually, this property is not universal, but depends on whether the TSA asked for undeniable identification of the requestor (e.g. time-stamped data must be digitally signed by the requester).

Basic verification provides the verifier the undeniable knowledge that data existed at a given time, and were not altered afterwards. Further mechanisms are needed to prove ownership, out of PKITS concerns. As well, further refinement is needed to prove that multimedia data were recorded after some instant of time.

Losing the time-stamp associated to a multimedia stream prevents verification by the receiver. Still the issuer may store evidence and show it on request.

Transforming multimedia objects may prevent verification by the receiver. This includes good- and bad-will transformations. Strict digital signatures (where changing a bit make a large difference) prevents any verification but on the original data, that might not be available or be too dear to retrieve. Weak time-stamping allows verification of modified data, at the cost of opening a window for fraud: it cannot be said any longer that data were not altered after time-stamping.

There are several items involved in a digital time-stamping service. Some of them must be enforced as precisely as possible, namely the time and the secret used to sign. So for linking information, party identification, and so on. The only component that might be relaxed is the multimedia content itself.

Within these concerns, there are several opportunities to carry time certificates:

1. off-line
   This is the simplest way: multimedia information is a separate object from its time-stamp certificates. This technique allows for separate distribution of multimedia information with small overhead, for separate distribution of time-stamp certificates without uncovering the multimedia information itself, and for distribution of both in order to perform verification. Multimedia information shall carry unique identifiers for each piece subject to time-stamping in order to correlate time-stamps during verification.

   This is the method to be preferred when information may need to remain concealed.

   This method is similar to the traditional document time-stamping: there are two separate objects.

2. on-line
   This technique uses information headers foreseen in most standard formats to store time certificates. This format simplifies verification process since association of data to certificates in natural, and does not require to relate two different objects.

This is the recommended method by default.

3. as a watermark
   This technique takes the binary representation of the time-stamp information, and incorporates it within the multimedia information in such a manner that it can be retrieved directly from the multimedia content itself.

   This technique is to be preferred when objects are moved into different support media, and there may be little change to find an appropriate place to store the time certificate. It may be useful as well when other services are required such as copyright enforcement.

4.6.1 Off-line Transport Of Time Certificates

Multimedia information is an object. Time certificate is a different object. Or both are parallel collections of objects (a stream and a list of signatures, respectively). Time certificates may be distributed even before the data that may remain concealed; this is typically done when proof of possession is needed before disclosing contents. Bandwidth is saved as well if few verifications are carried on by the verifier, and only the relevant certificates need to be distributed (this depends on the time-stamping method).

It is essential that a unique identification scheme is used to relate signed objects to time certificates.

4.6.2 On-line Transport Of Time Certificates

The syntax of most multimedia document formats enables the inclusion of application-defined information. In digital still cameras, for example, this information could include the place where the picture was taken, the date and time, and the name of the person who took the picture.

The MPEG-1 and MPEG-2 standard formats provide two mechanisms to embed application information in the bit stream:

- The Video Elementary stream in MPEG formats has a USER_DATA section where the applications can store arbitrary information.
- The MPEG system layer allows for an elementary data stream to be multiplexed synchronously with the audio and video streams.

The good news of this technique is that users do not need to worry about unique identifications, nor to worry about handling two objects: a single object is self-contained.

Tools handling multimedia objects may take care of format changes, copying and re-encoding the time certificates as needed. Furthermore, irreversible data transformers may issue a warning about the risk of losing the opportunity to verify the data any longer, or proposing a renewal cycle to extend the previous time certificate to the new contents.

4.6.3 Time Certificates As Watermarks

Digital watermarking is a wide term to denote a collection of techniques and processes that can be used to embed data directly into the media, rather than into headers or wrappers. This ensures that the information is maintained across different data formats.

The inserted data should be robust again attempts to remove it via anticipated manipulations (such as image cropping, noise addition, filtering, etc.). In order to embed time-stamps in multimedia objects, both visible and invisible watermarking techniques can be applied.

The time value may be visible or invisible, or both. It is not a secret, and it is only a matter of convenience to show it clearly or not. When contents is audio, a “visible” time value means a trailer that “reads” the time value. When dealing with images, time value may be overprinted, or the image may be slightly extended to accommodate a readable printing. Visible watermarks convey information to the receiver before it goes into a verification process.

In order to embed time-stamps in multimedia objects, both visible and invisible watermarking techniques can be applied. Since time-stamps are digitally signed statements of existence, the validity of the digital signatures should be checked; therefore, the watermarks should be embedded in a machine-readable way. Imperceptible watermarks are by definition machine-readable, since humans cannot even notice their presence; therefore, we will require that **both perceptible and imperceptible watermarks include a machine-readable part containing the digital time-stamp**. For images, this can be achieved inserting one- or two-dimensional barcodes into the watermark, as shown in the following figure:

The robustness requirements necessary for the embedding of time-stamps in multimedia objects are subtle: on a hand, it is desirable that the watermarks carrying the time-stamps survive certain manipulations, such as cropping or resizing of images, or echoing of sounds,; on the other hand, if a multimedia document suffers severe manipulations, the watermark should disappear, or otherwise the time-stamp would falsely certify the existence in a past time of a newly obtained digital document. Therefore, we will require that the **time-stamp watermarks survive simple transformations of the multimedia object, but may be eliminated by significant alterations to the original document**.

In other words: embedding a time certificate as a watermark is a good-will work carried on to help receivers. But ultimate proof of existence should be carried out on the original data.

A very important point with time-stamp watermarks is that the original, unaltered multimedia object will not be necessarily available; therefore, we will require that **the watermark extraction process accepts as input only the time-stamped object, without resort to the original multimedia document**. This is again for receiver’s convenience.

Of course, the time-stamp encoding watermark should not hide important parts of the multimedia object; therefore, we will require **the time-stamp watermarking scheme to be unobtrusive** (this will hold automatically if the watermark is imperceptible).

4.7 THE ROLE OF THE TSA

A key issue in the handling of multimedia information is to decide who plays this TTP role. The options are that the entity generating the multimedia information is itself a TSA, or that an

external TSA is used. In either case, there is a need for periodic synchronisation of TSA activities as analysed in [PKITS D3].

A third option to take into consideration is the use of tamper-proof time-stamping devices. These are special purpose units that include the following functional blocks in a sealed package:

- a real-time clock
- read-write storage
- ability to perform cryptographic operations
- access control

	pros	**cons**
External TSA	▪ TSA election may be agreed by the parties	▪ communication costs and delays
Embedded TSA	▪ locality avoids communication costs and delays	▪ trust depends on synchronisation
Tamper-proof device	▪ locality avoids communication costs and delays ▪ policy is fixed	▪ trust depends on auditing of design and manufacture ▪ limited storage facilities

4.7.1 External TSA

The TSA is far away from the multimedia source. This is the most conventional architecture to provide time-stamping service using a TTP with its own policy, in its own operation environment. This works fine with any of the previous time-stamping schemes, as far as there is no real-time requirement, and the delay introduced by the remote service provision may be accepted.

4.7.2 Embedded TSA

The TSA is pretty close to the multimedia source: it may be the same CPU, or a close computer devoted to time-stamping. This scenario is very interesting when real-time requirements do not allow for external communications, and when the connection to an external TSA is not permanent. An external TSA is always needed for periodic synchronisation according to the agreed policy, and a system audit is recommended to prevent fraud. When synchronisation is seldom performed, system auditing must be enhanced. More frequent synchronisation required weaker system auditing. There as window of risk when a dispute arises before a synchronisation cycle is performed.

Synchronisation grain may be chosen taking into consideration the semantics of the information being processed. If the time-stamped information is naturally broken into pieces to be used by receiver (e.g. films, sequence recording, ...), then it is convenient to synchronise in the semantic breaking points. However, if the production is continuous, there is no other option than periodically scheduled synchronisation stages. If there are several multimedia production activities being carried out simultaneously, without a clean common breaking point, a local hierarchy of TSA may be convenient, each TSA devoted to a channel. Each ground TSA operates independently from start to termination of its assigned unit, and synchronises with a master TSA at the beginning and at the end. The master TSA synchronises itself externally according to an agreed policy.

4.7.3 Tamper-Proof Devices

These devices may play a significant role to achieve the double goal of being trusted, and operate close to the multimedia operation centres. External synchronisation may still be required, but it may be carried out over long periods of disconnected operation.

These devices may be produced under strict security conditions, tested to conform to high security standards, and then operate in any environment. It might be a removable intelligent chip card (ICC), a circuit sealed into a steel box, a PC card, ...

Access is allowed to users for time-stamping, and for verification, as well as to know the remaining capacity of the on-line memory. Access may be free, or require previous registration of users: a master account may be needed to permit account management; a limited number of accounts may be created, and even removed before being used (ever used accounts may not be removed to enable proper operation of the linking protocol).

Due identification of the user of the service is important if local storage of certificates is provided, and to avoid service abuse (e.g. looking for a denial of service attack). Identification mechanisms shall be provided; there is a wide range of them:

- password
- HMAC using a shared secret (password)
- user certificates issued by a CA known to the device
- ...

The private signing key never leaves the secure envelope. Time certificates may be internally stored (besides being published), or no storage for certificates may be provided, depending on the level of service. If internal storage is provided, the device has a finite life determined by the exhaustion of internal memory. If no internal storage is provided, still the life is limited by the validity period of the signing key. Since the device includes its own internal clock, there is no need for external synchronisation, that may nevertheless be scheduled periodically for trust reinforcement.

Removable devices may operate for a while, storing issued time certificates. A simple synchronisation stage links two devices for continued operation of the service. Removed devices may be easily used in court as evidence.

For the receiver of the time-stamped information to be able to verify time-stamps, the signing key must have a public certificate that is made widely available, duly issued by a reliable CA.

A highly secure device may use the basic time-stamping protocol: just a digital signature (see [PKITS D3]). Cheaper devices shall use the linked protocol for higher trust, preventing device malfunction or security breaks.

4.8 STORAGE

Multimedia information becomes so bulky so often that it is not realistic to expect central sites to store other party data as evidence. Most likely, the interested party shall take care of its own storage facilities.

Many small production units may not have the facilities to store original data for long periods, being forced to delete it. Deletion may be complete (no remind), or just change the support media (e.g. digital records may be destroyed but a printed copy is filed). Even if the source is destroyed, there may be users retaining copies for personal interest (e.g. patients retaining their X-ray photographs for a life; families keeping historical video recordings for generations; etc.)

When long term storage does not imply lossy transformations, plain digital signatures over hash values are the best way to maintain evidence. This mechanism prevents the smallest modification.

When lossy transformations may occur (representation format changes, image manipulations, change of support media, ...) then it is required to use week hash values to maintain a probabilistic evidence. This mechanism prevents coarse modifications.

4.9 RENEWAL

Basic renewal process shall be carried on along the same guidelines as stated for raw data (see [PKITS D4]) in order to extend the validity period of the time certificate beyond identity certificates expiration, and mathematics progress as related to cryptographic algorithms.

It must be further noticed that

1. multimedia objects shall not be typically subject to more than one hash function, and there is no safeguard to prevent attacks that are able to find collisions
2. the usage of descriptors rather than blind hash values to characterise information opens a difficult to estimate window for new collision techniques; nevertheless, descriptors shall never be used in isolation, but several of them shall be always needed simultaneously
3. one-time signature techniques, as suggested for certain scenarios, are relatively new proposals for which there is little usage experience

In the case of multimedia information, renewal process may also be used to allow data to survive destructive transformations. Let I be original information with its time certificate T(I). If I becomes I', a new time certificate, T (I', T (I)), may be issued that includes a reference to T(I), effectively extending the value of T(I). A trusted agent may be needed to certify the similarity of I and I', either when renewing, or later on when the time certificates are used to sort out disputes.

4.10 TIME SERVICE

Usual requirements on secure time sources, as described in [PKITS D3] are needed for multimedia information time-stamping.

However, there are two issued of concern: location, and sequencing.

Real-time requirements to time-stamp large amounts of information may force to place the TSA close to the multimedia source, as described above. It may not be feasible to provide a TSA with strict security features as for specialised TSA in controlled environments. The TSA may have to work in cost-constrained equipment, or exposed to physical attacks. Regular synchronisation is the correct approach to survive poor TSA deployments. Still, when cost is a major concern, frequent communication may be too expensive, and it may be delayed for long periods of isolation. As soon as multimedia information is exchanged with other nodes, a synchronisation cycle should be performed.

On the other hand, multimedia streams have some self-sequencing properties that are secured after digitally signing the data. Combined signing of concurrent streams provides inter-stream synchronisation. Time-stamping schema based on chaining (either backwards chaining or linked one-time signatures) further enforce sequencing.

Internal synchronisation is as good as real time stamps as far as external data are not involved. To compare local data with external data, absolute time stamps are needed. It is not unusual that low-cost capture devices push their data into more powerful concentrators. Local time-stamping may be reinforced by combined time-stamping at these concentration nodes, that do not have to

transform and compress information (therefore there is processing power to perform digital signatures), and may be located in adequate (secure) locations. As the scheme is repeated several times, a hierarchy of time-stamping concentrators may be deployed before a high quality TSA is available for overall time-stamping.

5 REQUIREMENTS

The term "multimedia" refers to the use of a variety of media types such as video, audio, text, graphics, animations, etc. encapsulated alone or together in digital documents conforming a single piece of information. Along with this heterogeneous classification in types, each media type can be represented, traded and/or transported under different formats and standards. These facts force multitude of scenarios with different requirements in each one of them. Here we present general common requirements for time-stamping multimedia information (we could not talk properly about multimedia documents) although some requirements will be of vital importance (regarded as essential) in some specific scenarios of use.

5.1 USERS' REQUIREMENTS

5.1.1 The Requester/Verifier

The requester of a multimedia time stamping service is generally a producer or broadcaster of that information and its needs may vary in a wide spectrum depending on the kind and 'quality' of data to deal with. The requirements from the service requester are therefore heterogeneous. The verifier requirements for multimedia information perfectly match those of the requester. Note that verification could take place on-line, real-time, at the same output rate that the corresponding requester is submitting that info to the TSA. Here we present general basic requirements that are common to multimedia TSS provision.

1 Reliability.

The main requirement over TSS is reliability. The service must be provided as stated in specifications.

UR_1	The TSA **shall** provide means to prevent from an incorrect or deficient service provision, these regards every element involved in the provision of the service.

2 Availability.

The user of TSS demands a 7-24 service. The specific case of TS for continuous streams (e.g. surveillance) of multimedia information will demand a continuous non-stop service provision. This constitutes a strong requirement on availability. Security practices are the most valuable resource to keep the system available.

UR_2	The TSPS **may** state an availability rate of the system providing TS issue/verification service.
UR_3	The TSA **shall** provide means to avoid service interruption and keep its availability rate within limits established in its TSPS (if stated).
UR_4	The TSA **shall** assist in the verification of **any** time certificate issued by it.

3 Accuracy.

The accuracy of time assigned to documents may be of vital importance to the user. Most multimedia applications shall not accept low precision clocks, the inherently fast nature of multimedia information presentation/generation leads to high requirements over STS precision; real-time broadcasting, atomic applications, astronomical applications...

UR_5	The TSA **shall** specify the origin and nature of its time source/s in its TSPS.

UR_6	The TSA **shall** state the precision (or precision options in case there are more than one alternative) that can be expected in time stamping service according to the level of service agreed with the TSA. This **shall** be defined in the TSPS.
UR_7	The TSA **shall** periodically publicise information regarding the accuracy of its time source and inform of deviations along with corrective measures taken.
UR_8	The TSA **should** provide means in order to check the status of time source and detect deviations.

4 Security, integrity and authenticity.

The user may require security and integrity over the information exchanged with the TSA. User may also want to protect the communication against "man-in-the-middle" attacks by requiring authentication over TSA identity. To meet these requirements public key cryptography and digital signature schemes could be used although it could be a serious drawback for performance. Alternatives to achieve fast authentication are one-time password schemes (one password for one session) or dedicated secure channels between client and server.

UR_9	Each member of the UoU **should** be in possession of an identity certificate issued by a CA, and **should** use it to prove identity to the TSA.
UR_10	The TSA **should** use a secure digital signature scheme (and key size) to produce the time-stamping certificates.
UR_11	The TSA **should** use a secure digital signature scheme (and key size) to communicate with users regarding matters apart from time stamps production.
UR_12	The TSA **should** provide secure dedicated channels to communicate with requesters providing high input rates.
UR_13	The TSA **should** use a one-time signature scheme where time restrictions forces so.
UR_14	The TSA **shall** provide means to secure and publicise information regarding its service.
UR_15	The TSA **shall** not disclose to unauthorised third parties any private information related to the users of its time stamping service.

5 Speed & Transmission Rate Negotiation.

The speed of the system is essential when serving TSS to "multimedia" producers/broadcasters, this is specially important when dealing with infinite length streams and not even minimum accumulated delay is allowed. Multimedia information must usually be processed in real-time, this imposes requirements over equipment (see section 5.3) to perform time stamping operations much faster than input rate. In this sense the client will require a previous rate negotiation phase in order to state the resources dedicated to that client in the server not go under speed requirements during operation.

UR_16	The TSA **shall** provide means (processing power) to keep its response times within the limits stated in its TSPS.
UR_17	The TSA **should** periodically publicise the distribution of its response times.
UR_18	The TSA **shall** establish the input rate of multimedia information it will be able to deal with in relation to a given client. This **shall** be achieved by means of a previous phase in the protocol where the transmission rate is stated.
UR_19	The TSA **shall** state in its TSPS a "spare rate" indicating the percentage and frequency of frames that could be skipped in case of experiencing problems during transmission.

6 Evidence.

The requirements regarding storage of evidence are imposed from the verifier and included in TSPS. Verifier may also want to check temporal relationship among documents time stamped by different TSA's. So s/he may require evidences of synchronisation from a given TSA.

UR_20	The TSPS **shall** state the period in which the time-stamped evidences shall be maintained, published and finally removed.
UR_21	The TSA **shall** keep information (evidences) for verification purposes as stated in TSPS.
UR_22	Each member of the UoU **should** store the time-stamped documents in a safe place, since they will be needed for verification and certificate renewal.

7 Cryptographic quality.

Users trust will be strongly balanced towards the strength of cryptographic primitives used in the time stamping protocol. The TSA shall state hash functions which shall be accepted in the request's hash field, working with multimedia information forces a compromise between cryptographic quality/security and performance.

UR_23	The TSA **shall** select and state in the TSPS a set of acceptable hashing algorithms that users may use to produce the hashes of their documents when following the time-stamping protocol.
UR_24	Each member of the UoU **shall** use an accepted hashing algorithm when following the time-stamping protocol.

8 Cryptographic feasibility.

Cryptographic operations must comply with some strong timing constraints when working with multimedia information. A compromise must be chased between the digest size, the speed of the function and its collision resistance.

UR_25	The TSA **shall** select and state in the TSPS a set of acceptable hashing algorithms accepted at each input rate and representation standard.

9 Specification of acceptable standards.

The wide spectrum of standards used in the representation of multimedia information will force the TSA to publicise what standards shall be accepted to be time stamped.

UR_26	The TSA **shall** publicise in its TSPS those representation standards acceptable together with its approximate accepted input rate (if applicable).

10 Storage requirements & compression tools.

Multimedia information requires large amounts of data to be stored, the interested party (requester mainly but eventually the verifier) would hold the responsibility for its data storage. Compression/decompression tools is therefore imposed for storage and transmission. Note that certain multimedia information will be discarded once it has been displayed, in these cases verification should take place with data before it is discarded and therefore a strong timing requirement arises.

UR_27	Users **may** have resources to compress/decompress multimedia information.
UR_28	Users **should** have space to store their certificates and multimedia info.

11 *In-situ (Local) Time-stamping & synchronisation.*

Timing requirements make remote operation with a TSA unfeasible. This is so when real-time operation together with high input rates have to be reached. In this cases the TSA must be very close to the multimedia operation centre. This forces local time stamping of multimedia information and periodic synchronisation with an external TSA.

UR_29	The local TSA **shall** state a synchronisation policy to external TSA. Synchronisation policy **shall** include frequency, audit trail actualisation, linking with internal information (if needed)...
UR_30	The TSA **shall** issue a TSPS exposing its security countermeasures.

12 Minimum distortion.

When transformation (lossy compression, re-scaling, printing, broadcasting...) occurs to multimedia information due to processes required to time stamp it, minimum distortion on documents quality must be achieved. By no means, changes must be perceptible to human observers or recognition devices (voice recognition, image profilers...).

UR_30	Transformation needed to time-stamp multimedia data **should** not affect the representation of that data.

13 Tolerant Time Stamping (Indelibility).

The user may require that certain transformations over their documents do not affect some information embedded within it. This is achieved by the use of watermarks and/or the time stamping of invariant properties.

UR_31	The TSA **may** provide the certificate embedded within the data with resistance to certain transformations.
UR_32	The TSA **shall** state in its TSPS those invariants preserved when serving tolerant TSS and the formal identifiers of the corresponding feature extractors.

14 Legal validity.

The user may need his/her time certificate to be an acceptable evidence for its use in court. TSA must provide means to build evidences that support veracity and integrity of certificates. This may be simply impossible in streams of infinite length, for its storage is not viable.

UR_33	The TSA **should** make reference to legal authorisations and permissions to operate and limit its liability and legal validity of certificates within its TSPS.

15 Universality.

An important requirement over time stamping certificates may be its validity beyond the domain covered by the emitting TSA. In order to met this requirements TSA may establish links either via legal (cross certification) or via technological (synchronisation) towards other TSA's.

UR_34	The TSPS **shall** define links with other TSS and domain where its certificates are valid (and verifiable).
UR_35	The TSA **may** provide synchronisation means to establish links to other TSS enlarging the domain where its time certificates are regarded as valid.

16 Lasting certificates.

The user may require a long term for his/her certificates. Renewal process may be regarded as uncomfortable to the user.

UR_36	The TSA **shall** assist its users in the periodical renewal of its certificates.
UR_37	The TSA **shall** publicise among its users information on cryptographic advances that could affect the validity of their time stamping certificates.
UR_38	On cessation of its activities, the TSA **shall** transfer to another TSA all the information that could be needed to verify or renew the time-stamping certificates issued during its activity period.

5.1.2 The Provider (TSA)

We identify requirements imposed from the TSA either to protect its assets or to be able to provide a given service quality as specified. In this sense, requirements will be imposed on users and on itself.

1 Service provision and level of service

UR_P1	The TSA **shall** publicise a TSPS defining its service and therefore limiting its liability.
UR_P2	The TSPS **shall** state the duration of the pair of keys used to time-stamp. The key used to time-stamp messages must be changed every specific period or every specific number of messages time-stamped. This period of time and number of messages **shall** be stated in the TSPS.
UR_P3	The TSPS **shall** specify the protocol used in the TSS provision.
UR_P4	The TSPS **shall** specify the term in which the time-stamped evidences **shall** be maintained in the TSA.
UR_P5	The TSPS **shall** state the precision (or precision options in case there are more than one alternative) that can be expected in the time stamping service according to the level of services agreed with the TSA.
UR_P6	The TSPS **shall** state the source from which it obtains the time reference and the procedures to avoid deviations from this time base.
UR_P7	The TSPS **shall** specify hash algorithms which shall be accepted in the corresponding request field, as well as the formal identifiers of them.
UR_P8	The TSPS **shall** specify the origin and nature of unpredictable events within the link chain, as well as the number of unrelated bits used to encode these events.
UR_P9	The TSPS **should** make reference to international standards applicable to the Time Stamping Service Provision followed by the TSA. This item is important for the interoperability and synchronisation with other TSA's.
UR_P10	The TSPS **shall** state the communication channel to be used for the communications between the TSA and users.
UR_P11	The TSA **may** be provided with redundant access channels. The use of several accesses will protect the availability of the TSS.
UR_P12	The TSPS **shall** state how it shall proceed in case of going out of the Time Stamping Service Provision activity.
UR_P13	The TSA **may** encourage the use of more than one hash for the same document.
UR_P14	The TSPS **shall** state how all deviation from the normal operation shall be communicated to the customers and published to made them known to any interested party. Deviations includes loss of synchronisation, deviations from the time base, compromised keys, etc.

2 Security practice.

UR_P15	TSA **shall** use technical security controls to deliver its services in a secure way. Security control of the equipment, of the life cycle of the software, of the networks and of the cryptographic modules are included.
UR_P16	Each member of the UoU **shall** be in possession of an identity certificate issued by the CA, and **shall** use it to prove identity to the TSA.
UR_P17	The TSA **should** be provided with a special security length key for time stamp tokens signing purposes. The length of the key should be secure "enough" regarding the state of art for signature forgery.
UR_P18	The TSA signing key lifetime **should** be restricted to a given "short" period of time.
UR_P19	The TSA signing key lifetime **should** be restricted to a maximum number of digital signature operations.
UR_P20	Synchronisation procedure **should** be launched periodically and its frequency must be high "enough" to detect and correct possible deviations.
UR_P21	Access to time source receptor **shall** be strictly restricted and placed in a secure location.
UR_P22	Receptor location **shall** provide an stable signal reception and **shall** avoid shade or dark reception zones.
UR_P23	The TSPS **shall** state how and when audits are carried out and who must do it (external or internal).
UR_P24	The TSPS **shall** state what events shall be recorded, the type of log files, the procedure to do it, who is the responsible, and how long the log files shall be maintained.
UR_P25	The TSA **should** insert the hash of an unpredictable event every N time-stamps.
UR_P26	The TSA **should** insert unpredictable events periodically within its linking information.
UR_P27	The TSA **shall** state its internal and external procedures to recover after a disaster of any type has occurred.
UR_P28	The TSA **shall** use non technical security controls to deliver its services in a secure way. Physical, personnel, and procedural controls are included.

5.2 SOFTWARE REQUIREMENTS

This section refers to the specific software requirements for TS of multimedia information.

5.2.1 Cryptography

- ✓ Hash function.
  Users shall use one or more hash functions to calculate message digests that shall be sent to TSA. Sometimes, in the case of specific multimedia information, the use of a single hash function shall be mandatory due to time restrictions.

- ✓ Digital signature scheme.
  Requests and tokens shall be signed by the sender, therefore a digital signature scheme is needed.

5.2.2 Communications

- ✓ TCP/IP support.
  Actual communications are basically based on TCP/IP so support for these protocols is essential.

- ✓ Communication applications: Web browsers, E-mail clients, ftp apps,...
  Users may use some client applications for accessing to the Time Stamping Service.

- ✓ HTTP server, E-mail server, ftp server,...

TSA shall provide server application in order to respond to client service requests.

- ✓ Access to corresponding Authorities, CA's.
  Both the user and the TSA shall be provided with communication means to access services delivered by other authorities.

5.2.3 Data bases

- ✓ Requesters information storage.
  The TSA shall store this information for verification purposes.

- ✓ Evidences storage.
  The TSA shall store this information for verification purposes.
  Multimedia information may need to be stored as evidence for later processing, or discarded when the session terminates.

- ✓ Audit trails.
  The TSA shall store this information for audit purposes.

5.2.4 Security

- ✓ Software firewalls.
  The TSA shall use firewalls to restrict access to its infrastructures.

- ✓ Monitoring of network traffic.
  The TSA may use specific software for monitoring the use of its services.

5.2.5 Multimedia specific software

- ✓ Interactive multimedia applications.

- ✓ Multimedia editor.
  Multimedia information shall be analysed and decomposed to introduce the time stamp.

- ✓ Feature structure.
  When a multimedia file has been time stamped, users may desire that the specific transformations will not rend the time stamp invalid.

- ✓ Compressors and decompressors.
  Multimedia information usually involves huge amounts of data. The sample sizes are usually so large that compression algorithms are used to reduce them.

5.2.6 Other

- ✓ Trusted Operating System.
  The TSA shall use high security operating systems.

- ✓ NTP software.
  If the TSA is to use NTP as secure time source, it shall be provided with software to do so.

5.3 HARDWARE REQUIREMENTS

5.3.1 Real-Time Clock

The TSA shall own its own reliable clock as the primary source of time. Normal TSA operation requires read access to the clock. Eventually, the clock may need adjustments to synchronise

with reference clocks outside the TSA. Writing the value of the clock shall be a closely monitored action that requires specific authorisation, and the change is recorded. Changes become part of the time-stamped information the TSA deals with: if a linking protocol is being used, the log of the clock adjustment shall be one document in the linked chain. If another protocol is used, the log becomes part of the activity reports that are submitted to other TSA for synchronisation.

Clock source shall be enclosed with the digital signing machinery, and altogether secured under physical and logical security means of the TSA provider.

External clock sources shall be regarded as references that do not automatically modify the internal clock. When the difference between the internal clock and the external reference goes over the accuracy stated in the time-stamping practice statement that applies to the service provision contract, the measured difference shall be recorded into time-stamped operation activity logs, and the service administrator shall be notified to take appropriate corrective actions, as described above.

Clock accuracy shall be under the requirements stated in the most restrictive practice statement of the TSA. Temporal deviation shall fall within stated accuracy for periods so long as the foreseen synchronisation rate. That is, time drift between synchronisation shall be under contracted accuracy.

5.3.2 Processing power

- ✓ CPU processing power.
  TSA and users shall arrange processing power enough to deal with multimedia information.

- ✓ Cryptographic processing power.
  The TSA may be provided with specific cryptographic hardware units. This units provide fast cryptography, and enclose private keys under secure enclosing.

5.3.3 Storage requirements

- ✓ Hard disks.
  for operation activity logs, links storage, and evidence storage.

- ✓ Streamers.
  for regular back-ups, and long term (off-line) evidence storage. as well as to send evidence packs to other authorities under lawful request.

- ✓ Redundant storage equipment.
  RAID or mirroring, or any other mechanism shall be deployed to guarantee that evidences are never lost, within the liability stated in the contracts. Loss of data as a consequence of external storage malfunction may have serious consequences: economic penalties, image degradation, and loss of customers' trust. The damage is limited by periodic synchronisation with other TSA.

5.3.4 Communications

- ✓ LAN.
  TSA subsystems may be connected by means of local area networks. This internal network shall be behind the firewall, and subject to strict access control and activity log.

- ✓ Radio receiver.

To get time references from external sources, that shall be used to detect differences with the internal clock source.

- ✓ GPS receiver.

- ✓ Communication means.
  Multimedia information transmission needs communications lines with a large bandwidth. It is important to guarantee both a minimum data throughput, and a maximum transit delay for timely provision of the service.

- ✓ Communications management equipment.
  Communications equipment must be kept up and running according to needed performance. The most difficult issue is to deal with capacity overflows, either because of an underestimation of the needed resources, or as a consequence of denial of service attacks. Resource dimensioning needs continuous monitorisation to foresee long term trends and react to new needs. Quick detection of denial-of-service attacks permits some reactions (e.g. closing a channel, disallowing a source, shutting down a sub-service, etc.) that minimises the damage and keeps the overall service running.

5.3.5 Security

- ✓ Hardware firewall.
  The TSA shall use firewalls to restrict access to its infrastructures. Firewall configuration and operation shall follow the TSA security policy and security procedures that are in place to back the TSA contracts.

- ✓ UPS (Uninterrupted Power Supply).
  High quality service provision may require non-stop operation. Non-stop service requires a UPS able to feed systems until a secondary power supply is ready to replace normal power. Most typically, medium grade quality of service, shall require graceful system shutdown. For this service, the UPS only needs a few minutes to keep systems powered until internal process information is saved in permanent storage, and disks stop.

5.3.6 Other

- ✓ Internal management and invoicing equipment.

6 APPLICATION PROGRAMMING INTERFACE

This section will describe the concepts and architectural criteria to design an application programming interface to time-stamp multimedia objects. Currently, there is no universally accepted framework to program multimedia applications. We will base our description on the Java Media API that is under specification by Sun Microsystems as part of the Java Platform. The reason that has led us to choose this platform is its advanced level of abstraction, as well as its good positioning to become an industry standard. However, the needed Java interfaces for cryptography and multimedia are not finished, and still some effort is needed before concrete details can be fixed at the level of objects and interfaces. This section states the way to proceed.

6.1 JAVA MEDIA

Java Media is a set of application program interfaces designed to support multimedia in the Java platform. These are the following:

API	**Scope**
Java 2D	Two-dimensional graphics and image manipulation
Advanced Imaging	Advanced, high-performance image manipulation
Java 3D	Three-dimensional graphics runtime and three-dimensional sound
Java Media Framework	Playback of synchronised multimedia streams
Java Collaboration	Media data sharing
Java Animation	Two-dimensional animations
Java Sound	Sound processing and MIDI synthesiser
Java Speech	Speech analysis and recognition
Java Telephony	Computer-Telephone integration

The most elaborated Java Media APIs are Java 2D, Advanced Imaging, 3D and Speech. We will consider the time-stamping of Java 2D and Java Sound multimedia objects; the treatment of the remaining multimedia types will be very similar.

6.2 JAVA CRYPTOGRAPHY ARCHITECTURE

The Java Security API is specified by Sun Microsystems as a new Java core (standard) application programming interface which provides cryptographic objects and operations. The first release of this API includes functionality for digital signatures and message digests, as well as key and certificate management and access control. An extension to this API, the Java Cryptography Extension includes encryption and key exchange; this extension is publicly specified, but will not be released outside the United States due to export control of cryptographic material.

To perform the time-stamping of multimedia objects we will make use of the following classes:

- *MessageDigest*

  This class is designed to provide the functionality of cryptographically secure message digests (also called *hash functions*), such as SHA-1 or MD5. The time-stamping classes make use of *MessageDigest* in the creation and verification of time-stamps.

- *Signature*

  This class provides the functionality of a digital signature algorithm such as DSA or RSA-with-MD5. This class is used in the generation of the digital signatures that are contained in the time-stamps.

6.3 JAVA OBJECTS USED IN THE TIME-STAMPING API

We will define some common interfaces and classes to perform the time-stamping of the different multimedia types:

- *TimeStamp*
  This class will represent a time-stamp, that is, a digitally signed (either with a traditional or a one-time digital signature scheme) certificate of existence of a certain multimedia information sample in a specific point in time. The *TimeStamp* class contains fields for the time information, for information on the multimedia object to which the time-stamp refers, the identity of the time-stamper, and the digital signature.

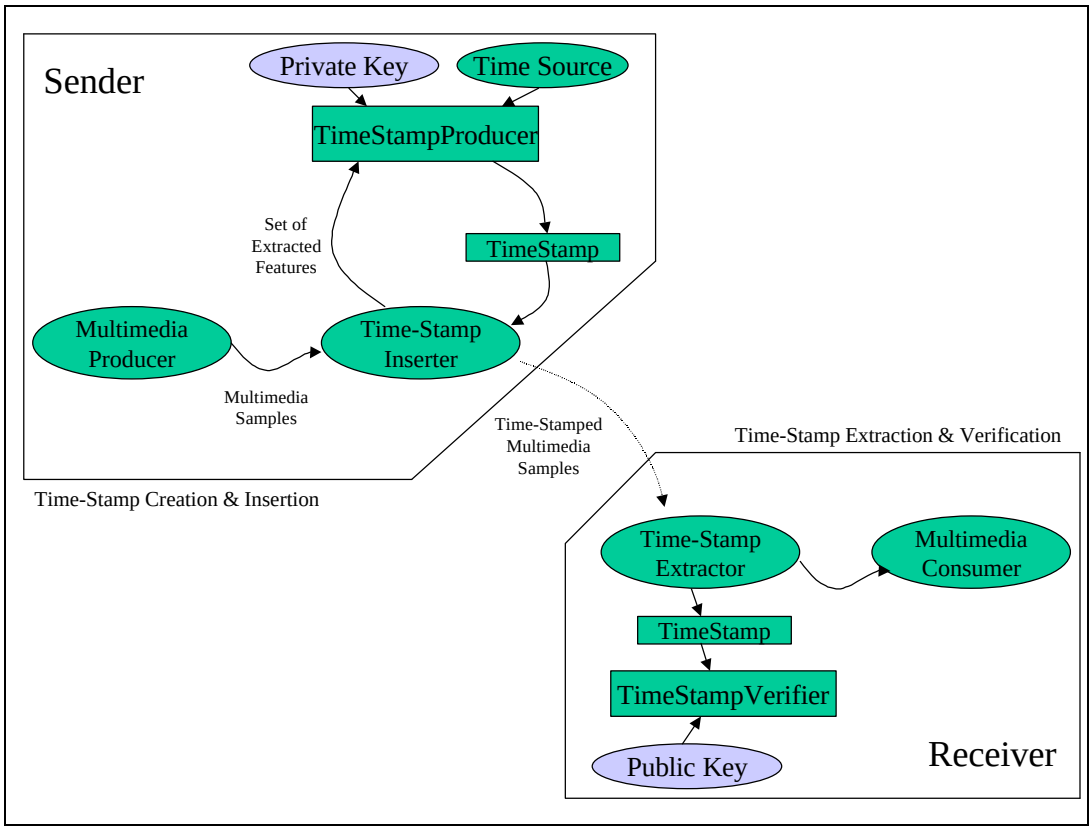
- *TimeStampProducer*
  This interface defines a single function, *getTimeStamp*. This function accepts information on a multimedia object, and returns the corresponding *TimeStamp* object. Depending on the kind of time-stamping performed, this information will be a hash (or set of hashes) or a set of features extracted from the multimedia object. On initial setup, this object receives information on the time-stamping entity, and of course its *PrivateKey* (class defined in the Java Security API) object.

- *TimeStampVerifier*
  This interface also defines a single function, verifyTimeStamp. This function accepts as argument a TimeStamp object, and returns *true* or *false* according to whether the time-stamp is valid or not. On initialisation, the PublicKey (defined in the Java Security API) object corresponding to the signer must be provided.

- *InvalidTimeStampException*
  This exception will be raised by objects defined below in order to indicate that a received multimedia object contained an invalid time-stamp.

6.4 API FOR TIME-STAMPING OF MULTIMEDIA OBJECTS

The API for time-stamping of multimedia objects fits naturally in the Java Media APIs. The following figure describes the overall architecture:



The steps are the following:

1. An object is producing the multimedia samples; these samples can be obtained from a local or remote file, from a live capturing device, etc.
2. A multimedia filter extracts the selected features from the multimedia object.
3. The multimedia sample features are passed *TimeStampProducer* object, which inserts the time information and performs a digital signature, thus producing a *TimeStamp* object.
4. The multimedia filter inserts the time-stamp in the multimedia sample, either adding the information taking advantage of the special media format, or using digital watermarking techniques.
5. The multimedia information travels from sender to receiver as usual.
6. The received multimedia samples are received by a time-stamp extraction filter, which analyses the incoming samples and obtains a *TimeStamp* object from them.
7. A *TimeStampVerifier* object checks the validity of the time-stamp information. If the time-stamp is invalid, the extractor filter will raise an *InvalidTimeStampException* to inform the higher application.

This scheme will be used for all media types in the Java Media API.

6.4.1 Time-Stamping of Still Images (Java 2D)

Java Media defines an object model that can be used to manipulate and display both still-images and image sequences. The object model is defined in terms of an abstract class, *Image*, and three interfaces, *ImageProducer*, *ImageObserver* and *ImageConsumer*:

- *Image*
  This class represents a platform-independent image. It has properties such as height, width, bit-depth.

- *ImageProducer*
  This interface contains functions for an object capable of creating Image objects. From the *ImageProducer* interface it is possible to obtain an Image using one of the *createImage()* methods; from an *Image* object it is possible to reach its associated *ImageProducer* calling the *getSource()* method.

- *ImageObserver*
  This interface defines one method, *imageUpdate*, is called to indicate that an operation on an Image object has finished, also providing some information on the type of changes that the image has suffered. When the object implementing the *ImageObserver* interface returns *true* to the imageUpdate call, it receives further information on the image; if it returns *false*, the *ImageObserver* will not be called when further changes to that image take place.

- *ImageConsumer*
  This interface contains functions that enable an object implementing it to receive Image objects or portions of them.

  There is also a very important class, *ImageFilter*, which implements the *ImageConsumer* interface in order to receive the contents of the images to be processed. We can describe an *ImageFilter* as an *ImageConsumer* that manipulates the incoming image data and forwards the result to another *ImageConsumer* class.

6.4.2 Time-Stamping of Sound (Java Sound)

In Java Sound there is for the moment no provision to handle audio capture; therefore, we will assume that the sound information will be obtainable from a certain Uniform Resource Locator, and that it has been time-stamped at the time of capture. We will make use of the *AudioPlayer* class. This class encapsulates the functionality needed to play audio streams, and this is the appropriate encapsulation for a time-stamp verifier: the audio stream is received, and our time-stamp verification *AudioPlayer* object will verify the time-stamps attached to the incoming sound samples. If an invalid time-stamp is received, the object will raise an *InvalidTimeStampException* exception to inform the higher levels.

7 SCENARIOS OF USE

This section surveys usage situations and proposes concrete techniques for each one. It is one level more of concretion of the general ideas covered along the document.

7.1 CD-ROM DISTRIBUTION

In this scenario, a large bunch of information is packed together into a single transport device. There is plenty of time to process the material before distribution, and data are available to be randomly accessed. In this situation, the preferred technique to time-stamp the data is either as a whole or using a table of hashes. Time-stamping as a whole makes sense when there is a single time to be referenced. Using a table makes sense when there are different time references for different parts of the enclosed material. Random access to the hash table allows for efficient verification of single time certificates.

If the material provided in the CD-ROM is to be repacked, the proposed data structure becomes useless. Several options are available to fragment trust:

- apply a renewal protocol, issuing time-stamp certificates for single parts and including the time certificate of the whole, to extend trust
- use a protocol that signs blocks in isolation as the native protocol to sign CD-ROM contents (there is a significant data overhead)

7.2 BROADCAST

Broadcast radio and TV pose their specific requirements for time-stamping. There are two basic situations: live material, and stored material.

Live broadcasting is pretty usual. For instance, sport events are typically broadcasted with a small delay, although they may be replayed later on. Another interesting situation is ads accounting: it is of paramount importance to know exactly when was an ad inserted, since prices change dramatically.

Each ad is a clear block to time-stamp in isolation. Time-stamping must be fast, since resolution is about seconds. On the other hand, verification is performed much later, and there is no strong requirement to be fast or cheap. Since the ad is well-known in advance, the hash value of the contents is previously evaluated once, and the clock is added every time the ad is performed.

For live transmission an on-line time-stamping technique without long look ahead is needed. Furthermore, the signing and verification processes have to be quite light. A one-time signature scheme is appropriate. This approach permits to time-stamp once either for immediate transmission, and to store for later replay. However, if the material is to be used frequently in the future, it may be convenient to time-stamp again using another method that is more effective, e.g. a table of hashes. The list of hash values and time values may be prepared in real time, and signed as a whole upon termination, or introducing breaking points (say every 60 minutes). One-time signature permits the real-time receiver to verify time-stamps on the fly, while the table of hashes permits easy storage for random access in the future.

For stored material to be broadcasted, look ahead chaining is very convenient to permit the receiver on-line verification without the overhead of storing a large buffer of information. A renewal phase may be convenient to transform from the table of hashes into the chain of hases.

It is pretty likely that the time value as well as the originator identification is added to images as visible watermarks.

Traditional TV broadcasting (analog) can do little more than providing visible watermarks, and the receiver TV-set is unable to carry out a verification process.

Modern digital TV requires processing power on the client side (the decoding and decrypting set-top box) that may verify time-stamps and present the outcome overprinted on the TV set. These boxes are rather limited in terms of memory and processing power. Therefore, light weight mechanisms are needed. Again, chaining is to be preferred.

7.3 VIDEO-CONFERENCE

This is a case of live material that involves two or more parties (therefore, two or more synchronous streams). Material may need to be stored as evidence for later processing, or discarded when the session terminates.

If the material is to be discarded, time-stamp is most likely meaningless: it is real time, and each party knows it.

If images and audio are to be stored as evidence, time-stamping makes sense. Reliable time-stamping is needed for later replay. Furthermore, the different streams need to be synchronised, what means that the hash values on each stream must be periodically included within the signature of every other stream.

There is a question as whether the material may be time-stamped by the issuer (each one its own production) or by the receiver (each one each others'). These options correspond to a "proof of origin" and a "proof of reception", respectively; still there is no room for a full blown protocol to link both proofs. The problem in a video conference is that the human value is built on what is actually received, while different participants might receive different material in free format products. If there were a central distribution point that guarantees a unique stream for every one, this will be the time-stamping place. If there are as many points of view as participants, both time-stamping in origin and in reception may be needed: that implies that there will be several TSA working in parallel, that must synchronise from time to time.

7.4 WORLD-WIDE WEB (WWW)

The WWW is becoming a standard de facto to distribute and access multimedia objects. Web distribution includes text with formatting instructions (HTML), hyperlinks (e.g. URL), images (e.g. GIF), audio (e.g. Wav), and video (eg. MPEG-2); as well as active components (e.g. Java). It is important to time-stamp both each building component as its composition into a single document.

Each component shall be individually time-stamped, pictures, audio and video. Textual content (HTML) is normalised (in Internet this means CR-LF line terminators). The container (HTML) is hashed, including MIME information (e.g. local alphabet, encoding, ...) and the whole packet is time-stamped. The PKCS-7 time certificate is added as a new header to the HTTP exchange (most likely encoded in base 64).

HTTP and HTML define large collections of headers to convey useful information to the client. Only those that have an impact on content interpretation shall be subject to hashing for time-stamping. Still, these headers may appear in any order, and their syntax definition is quite loose. To avoid misinterpretations, and permit verification, the relevant headers will be replicated in the text (explicitly) for the hash function to depend on them. This trick does not require any convention on header usage, and opens the door for easy consideration of more headers in special situations in the future.

7.5 ELECTRONIC MAIL (E-MAIL)

This kind of material is quite similar to web material, although there is an important difference: a mail has typically a well defined sender. For this reason, this situation has already received some attention from the normalising bodies. Namely, there exist a de facto standard S/MIME (Secure / Multipurpose Internet Mail Extensions [rfc 2311]). In particular, this standard provides message integrity and non-repudiation of origin using digital signatures. Currently, work is under progress in IETF to further standardise this scenario.

S/MIME defines a procedure for unambiguously derive a PKCS-7 enveloped data package from a collection of objects encoded using MIME [rfc 2045, 2046, 2047, 2048, and 2049].

When the originator of the message is a TSA itself, the signing and the time-stamping procedures are merged together. [rfc 2311] describe the required and recommended attributes related to the message, and [PKITS D4] the attributes required for the time-stamping service.

When the message originator requires an external TSA to time-stamp a message, two digital signatures are to be carried along with message contents. The TSA is provided with PKCS-7 enveloped data that may be encrypted (or not), and is signed by the requester of the service. The TSA issues another PKCS-7 according to [PKITS D4] recommendations, where the signed data is the has value of the submitted PKCS-7 object (for blind time-stamping) or the PKCS-7 object itself (time-stamping clear documents).

When the time-stamping service is requested from someone different from the originator, the TSA shall handle the data as raw data [PKITS D4].

7.6 MEDICINE

There are many kinds of medical information that may be regarded as multimedia. For instance:

- radiographs (still images)
- scanner results (video)
- vital constant registries (e.g. heart beats, blood pressure series, ...)

Disputes related to patients in the future may require reliable time-stamps to use evidences in court, as well as a to correlate data that come from different (unrelated) sources, and time is the unique sound reference.

Still images on digital media match fine with a whole data time-stamp. One by one.

Video recording on digital media match fine with the same protocols that apply for CD-ROM: either digitally time-stamping as a whole, or using a table of hashes. The same criteria applies to data series.

One frequent transformation of medical records is to change the media: radiographs are printed, series of data are graphed, and so on. These transformations at least introduce two changes:

- resolution changes
- digital format is lost

For delivering the time, a visible mark overprinted to the paper copy may suffice.

For enforcing the association of the time information to the certifying TSA, a digital signature is needed, and it has to be ported to the new support media. Furthermore, the information to time-stamp must be something that can be regenerated from the new media, even if it gets older and parts of it were damaged, or even lost.

In order to survive the mentioned transformations, appropriate descriptors shall be used.

For grey scale images (as traditional radiographs) the following descriptors apply:

- Average value of all pixels in the image
- Intensity histogram
- Relative intensity histogram
- Normalisation with respect to size
- Highest-energy DCT / Fourier coefficients
- Lower DCT / Fourier coefficients

A radiograph shall typically carry a visible time value, at least for classification and retrieval. It shall further provide information about the descriptors used, their value, and their digital signature. Human oriented information may be visible (printed along the image). All these values, plus the digital signature shall be provided as a [bi-dimensional] bar code. Verification implies a rescanning, an evaluation of the descriptors, and a correlation analysis with respect to the stored values. Verification outcome is an index of coincidence that shall be accepted or not depending on the applicable policy. The digital signature shall be used to certify the values of the descriptors that are used as a reference.

Colour is used in many medical images to convey more information. Typically, it is not real colour, but a convention to carry information. When a few colours are used to denote a few characteristics, another parameter to take into consideration is the colour table. Furthermore, when the image is digitised for verification, colours shall be adjusted to the explicit table.

If colouring is used to carry analogic information, there is still an opportunity to normalise intensity before extracting descriptor values.

7.7 DIGITAL CARTOGRAPHY

Real state is a traditional source of human disputes. Industrial catastrophes may be caused by large engineering works (e.g. ponds, land organisation, ...) Reliable knowledge of timely information about earth surface is quite often used in courts. This may include photographs (e.g. taken by aeroplanes) or engineers' drawings.

Photographs require a treatment quite similar to that described above for medical records. It is important to take into account that these photographs quite often cover zones of the spectrum out of the human vision capabilities. Time-stamping must also include wavelength information to avoid attacks that aim to apply correct information to false wavelengths.

Engineering drawings are a different kind of material, since it uses typically vectorised representations of the data. This is an extremely compact storage technique that permits many transformations without degradation (e.g. scaling). Time-stamping a vector representation of data is very robust for good-will transformations.

Old plans may have no digital version, and they need preliminary scanning and vectorisation. The computer readable vector version may be time-stamped. The vector representation, its signature, and the time mark, may be printed out as 2-D bar codes to add to the printed plan for later verification.

7.8 DOCUMENT EXISTENCE NOTARISATION

This scenario occurs when a notary is requested to sign a multimedia object to be used as evidence, or when the public administration acts as a notary signing the reception of a multimedia object submitted by a citizen or enterprise (e.g. a bid). The most typical case are maps or plans: 2-D pictures.

If the map is digitised, the best approach is to add a time-stamp as a whole. The original document may be stored as added evidence, or only the hash value of it.

If the material is not digitised, a scanner is needed. There shall be a policy about scanning parameters that apply: a scale of, to adapt to different situations. Once the document is in digital format, a collection of descriptors is needed. Again, there shall be a previously fixed policy or collection of policies. The descriptors are evaluated, time-stamped, and stored as evidence. The submitter receives as well a copy of the descriptors, either in digital format or using some machine readable print-out. The objective is to store evidence enough to verify the document if it were later digitised again.

7.9 TELE-SURVEILLANCE

This is the scenario where data capturing devices are spread around to monitor a zone or industrial installation. There are typically video recorders, and sensors, that altogether provide a report either for real time surveillance or for analysing incidents in the future.

A tele-surveillance system is usually built out of a large number of remote recorders and a central station. The remote units work in difficult conditions: exposed to physical attacks, limited processing power, reduced to minimal cost, and not necessarily on-line. The task of remote units is to capture data, store it locally (e.g. for 15 days), and communicating from time to time with the central station to report incidents and download records. In order to reduce storage and transmission, the multimedia information is typically compressed in the remote unit.

When there are problems (e.g. robbery, catastrophes, ...) the recorded material becomes a mostly relevant evidence to analyse what happened, and act accordingly. Acting may imply determining responsible people, and may have legal consequences. Therefore, it may be quite meaningful to have trusted records, both with respect to data integrity, and time of occurrence.

The embedded TSA described in section 5.7.3 are excellent devices to deploy along remote units for local time-stamping. The system may work in isolation for long periods, and perform a synchronisation cycle whenever the central station if connected. Regular connections may be scheduled to impose a maximum period of working in isolation. The embedded TSA may store the time certificates it issues to add probing value to the reported material when this is needed in court to sort out disputes.

8 APPENDIX A: CRYPTOGRAPHY

8.1 EFFICIENCY OF CRYPTOGRAPHIC PRIMITIVES

8.1.1 Efficiency of Hash Functions

The following table presents the speed benchmarks obtained [Weidai] for the most popular cryptographic hash algorithms. All the algorithms were coded in C++ (no assembler) and compiled with Microsoft Visual C++ 5.0 with the "optimise for speed" and "Pentium Pro code generation" settings, and run on an Intel Pentium II processor executing at 266 MHz under Microsoft Windows NT 4.0.

Hash Algorithm	***Hash Size (bits)***	***Performance (Bytes / s)***
MD5	160	32.411.914
SHA-1	160	14.399.885
HAVAL (pass=3)	128	24.312.602
HAVAL (pass=4)	128	17.215.678
HAVAL (pass=5)	128	13.534.447
Tiger	128	8.520.408
RIPEMD-160	160	13.603.034

There is another performance benchmark of MD4-like hash functions by the authors of RIPEMD-160 (H. Dobbertin, A. Bosselaers, B. Preneel); the measurements are performed on an Intel Pentium processor at 90 MHz, with assembly implementations [BossGov+96], and executing in 32-bit flat memory model. The following table presents the comparison:

Hash Algorithm	***Hash Size (bits)***	***Performance (MBytes / s)***
MD4	128	23,90
MD5	128	17,09
RIPEMD	128	12,00
RIPEMD-128 (Update of RIPEMD)	128	9,73
RIPEMD-160	160	5,68
SHA-1	160	6,88

8.1.2 Efficiency of Digital Signatures

The following table presents the speed benchmarks obtained [Weidai] for the most popular digital signature schemes. All the algorithms were coded in C++ (no assembler) and compiled with Microsoft Visual C++ 5.0 with the "optimise for speed" and "Pentium Pro code generation" settings, and run on an Intel Pentium II processor executing at 266 MHz under Microsoft Windows NT 4.0.

Algorithm	*Number of Bits*	*Signing Time (ms)*	*Verification Time (ms)*
RSA	512	8,13	0,38
RSA	1024	46,01	0,99
RSA	2048	291,26	2,97
ElGamal	512	5,77	7,08
ElGamal	1024	24,83	29,91
ElGamal	2048	109,09	132,02
EC over GF(p)	168	40,82	49,50
LUC	512	18,17	0,58
LUC	1024	90,94	1,56
LUC	2048	541,07	4,70
DSA	512	7,86	9,43
DSA (with precomputation)	512	4,41	22,81
DSA	1024	24,47	30,30
DSA (with precomputation)	1024	9,54	20,26

8.2 ONE-TIME SIGNATURES

One-time digital signature algorithms are digital signature schemes that can be used to sign, at most, one message. If more than one message is signed with the same signature parameters, an attacker could very simply forge digital signatures. Therefore, new public and private keys are needed to sign & verify each message. An important advantage of one-time digital signature schemes is that their performance is, in most cases, much better than traditional digital signatures.

The most important one-time digital signature schemes are Rabin's,
(Goldwasser, Micali, Rivest). We will describe the most practical among them, Merkle's one-time digital signature; Rabin's and GMR are described in [Menez+97].

Merkle's One-Time Digital Signature Scheme

- ***Key Generation:***

  If Alice wishes to signe messages of length N bits, she must select a suitable hash function H and generate $t = N + log_2(N) + 1$ validation parameters, as follows:

  1. Choose t random secret strings, each of length l bits: $k_1, k_2, ..., k_t$.
  2. Compute the hash of each secret string: $v_i = H(k_i), 1 £ i £ t$.
  3. The public key (used to sign the message) will be $(v_1, v_2, ..., v_t)$.
     The private key (used to verify the signature) will be $(k_1, k_2, ..., k_t)$.

- ***Signature Generation:***

  To sign a message m of length N bits, Alice will follow these steps:

  1. Compute c, the binary representation of the number of zero bits in the message m (represented as a bitstring).
  2. Concatenate the message m and the computed c, to form $w = m \| c = (a_1, a_2, ..., a_t)$. [This is bitstring m followed by bistring c.]
  3. Determine the positions in bistring w taking value 1; that is, find the values $i_1, i_2, ..., i_u$ such that:
     $i_1 < i_2 < ... < i_u$, and
     $a_n=1 \Leftrightarrow n=i_j$ for some $1 £ j £ u$.
  4. Let $s_j = k_{ij}$, for $1 £ j £ u$.
  5. The signature of message m is $(s_1, s_2, ..., s_u)$.

- ***Signature Verification:***

  To verify Alice's signature $(s_1, s_2, ..., s_u)$ for message m, Bob would do the following:

  1. Obtain Alice's public key $(v_1, v_2, ..., v_t)$.
  2. Compute the bitstring c corresponding to the binary representation of the number of zeros in the message m.
  3. Form the concatenation $w = m \| c = (a_1, a_2, ..., a_t)$ of m and c.
  4. Determine the positions in bistring w taking value 1; that is, find the values $i_1, i_2, ..., i_u$ such that:
     $i_1 < i_2 < ... < i_u$, and
     $a_n=1 \Leftrightarrow n=i_j$ for some $1 £ j £ u$.
  5. Accept the signature if $v_{ij} = H(s_j)$, for each $1 £ j £ u$. Otherwise, reject it.

9 APPENDIX B: DIGITAL WATERMAKING

Digital watermarks are a relatively new development, originating as recently as 1993. In this appendix we will describe what digital watermarking is, how they work, and a number of proposed applications. We will also discuss their appropriateness for embedding of time-stamps.

9.1 INTRODUCTION TO DIGITAL WATERMARKING

"*Watermarking*" is the name given collectively to a set of techniques that enable the insertion and extraction of arbitrary digital information (called ") into digital multimedia objects.

The following figure illustrates the encoding (insertion of the watermark) and decoding (extraction of the watermark).

In the left figure below, a visible watermark indicates the origin of the map (http://www.city.net); the figure on the right contains a message embedded in an invisible watermark, which can be read with an appropriate software (http://www.digimarc.com):

Digital watermarking techniques can be classified according to whether the resulting multimedia objects are perceivable by humans:

- *Perceptible watermarking*
  The insertion of the watermark produces a noticeable distortion of the multimedia object, which can be perceived by humans.
- *Imperceptible watermarking*
  The insertion of the watermark produces a multimedia object that is indistinguishable from the original ("unwatermarked") one.

Perceptible watermarking is quite straightforward: a part of the multimedia document is modified or substituted by the desired information, which is encoded as a multimedia sub-object (sub-image, additional sound, etc.) of the same kind.

9.2 EXAMPLES AND APPLICATIONS OF DIGITAL WATERMARKING

Many watermarking algorithms have been proposed in the literature. Some techniques modify spatial or temporal data samples, such as in [vanSch+94], [Pitas96]. Other methods modify transform coefficients, such as in [CoxKil+97], [GiHar97]. We will review some watermarking schemes that are used to embed data in digital images:

- **"Least-significant-bit"**
  The simplest watermarking procedures are those that modify the least-significant-bit (LSB) of the pixels in the image. One such technique was proposed in [vanSch+94]: the information is a pseudo-noise sequence of pixels, encoded in the LSB of the pixels. Extraction of the watermark will be easily performed traversing the sequence of pixels and considering the LSB of the visited pixels.

  This method is not robust against intentional removal of the watermark, which is easily accomplished randomly manipulating the LSB of the pixels in the image; despite this, the watermark would survive some cropping and rescaling operations.

  This mechanism can be modified in a simple way by embedding the information in the second, third, etc. least significant bit. [Kurak+92] mention that even when the four least significant bits of an image have been modified, the eye cannot usually tell the difference. (This does not happen in audio information, where alteration of even the least-significant bit is noticed by the human auditory system.)

- **Patchwork**
  This technique is more sophisticated than LSB, and was proposed in [BeGru+96]. A set of *n* randomly selected pairs of image pixels is selected, $\{(a_i,b_i): 1 \le i \le n\}$; the intensity value of each $\{a_i\}$ is incremented in one unit, and the intensity value of each $\{b_i\}$ is decremented in the same amount. If for a given image we compute

  $$\sum_{i=1}^{n}\left[\ Intensity(a_i) - Intensity(b_i)\right]$$

  the expected value will be

  0 if the image does not carry the watermark
  2n if the image carries the watermark

  As we see, the watermark only carries one bit of information, and it modifies *2n* pixels in the image; therefore this technique presents the problem of being of very low capacity. Despite this, patchwork's watermarks are very robust again intentional or unintentional removal.

- **Texture Block Coding**
  This low-bit-capacity method is implemented by copying a region from a random texture pattern found in the picture to an area that has a similar texture. This results in a pair of areas in the image with identical texture, that can be detected via autocorrelation.

  Experiments mentioned in [BeGru+96] texture regions of 16x16 pixels can be decoded even after the picture has suffered a combination of filtering, compression and rotation.

  A major drawback with this method is that, although decoding is easily performed in an automatic way, encoding requires a human operator to choose the source and destination texture regions.

- **Transform Coding**
  Transform techniques insert the watermark in the image after a suitable reversible transformation, usually the discrete cosine transform (DCT). One of the first and most studied schemes is the one presented by Cox, Kilian, Leighton and Shamoon [CoxKil+96]. In this method the watermark is a sequence of real numbers $\{w_i: 1 \le i$

$\leq N\}$ obtained from a Gaussian distribution; these numbers are embedded in the N most significant DCT coefficients of the image multiplying the original coefficient by the amount $(1+\alpha w_i)$, where α is a suitably chosen scaling parameter, $\alpha \cong 0.1$. The extraction process requires the original image.

This watermarking scheme is very robust, and survives rescaling, JPEG compression, dithering, clipping, and printing-rescanning.

Let us describe some proposed potential scenarios for application of digital watermarking [MiBra+97]:

1. Perceptible watermarking for copyright protection
   Multimedia objects are made publicly available (e.g., via the Internet), and the content is concerned that the images can be used commercially without payment of the corresponding royalties. Here, if a perceptible watermark is inserted, any commercial use will have to eliminate the watermark; on the other hand, the multimedia information can be used for other purposes (research, education...). This approach is taken by most cartographic services on the Internet.

2. Visible watermarking to indicate ownership of originals
   In this scenario, the multimedia objects are also made publicly available, and the content owner wishes to indicate the ownership of the materials. Here the purpose of the watermark is to indicate the source of the information; loss of revenue is not a concern.

3. Invisible watermarking for trustworthy multimedia capture device
   In this scenario, a digital capture device (v.g., a digital camera) inserts an imperceptible watermark at capture time. The presence of the watermark at a later time will prove that the image was not altered since it was captured.

4. Invisible watermarking to indicate alteration of images stored in a digital library
   In this scenario, the owner of a publicly available digital library wishes to detect the alteration of multimedia documents without the need to compare them with the originals. The user will extract invisible watermarks from the documents, that will indicate whether that document has been modified or replaced since it was entered in the public library.

5. Invisible watermarking to detect misappropriated images
   In this scenario, the seller of multimedia documents is concerned that the documents may be purchased by an individual that will re-sell them at a lower price or for free. In this case, a “web crawler” is desired that would search images on web sites to look for the seller’s watermark automatically.

6. Invisible watermarking to indicate ownership
   In this scenario, the seller of multimedia documents suspects that one document has been edited and published without payment of royalties; presence of the seller’s watermark will indicate that the published document is property of the seller.

7. Invisible watermarking to indicate identity of traitor (“digital fingerprinting”)
   In this scenario, the seller of multimedia documents inserts a different watermark for each of the distributed documents, that indicates to whom it was sold. If an image is found published or distributed without payment of royalties, extraction of the watermark will indicate the identity of the misappropriator.

This scenario also applies to the distribution of secret documents: if a traitor distributes the document, extraction of the watermark will reveal the identity.

8. Invisible watermarking to command digital VCR
   In this recently proposed scenario, an invisible watermark is inserted in an MPEG-compressed video. The digital VCR will look for a special watermark that will indicate whether the video may be copied or only played.

9.3 CLASSIFICATION OF WATERMARKING TECHNIQUES

We will classify watermarking schemes into three groups: perceptible watermarks, fragile imperceptible watermarks, and robust imperceptible watermarks.

9.3.1 Perceptible watermarks

Application scenarios 1 and 2 share some desired properties:

- The watermark is readily perceptible.
- The watermark is unobstrusive.
- The watermark is hard to remove without severe degradation of the carrying multimedia object.

9.3.2 Fragile imperceptible watermarks

Application scenarios 3 and 4 share some common desired properties:

- The watermark is imperceptible to the human observer.
- The watermark is easily altered or removed by the application of transformations to the multimedia document, but survives to image cropping.
- It is difficult for an unauthorised person to insert a false watermark.
- An authorised person can readily extract the watermark.

9.3.3 Robust imperceptible watermarks

Application scenarios 5, 6 and 7 (and perhaps also 8) above share the following desired properties:

- The watermark is imperceptible to the human observer.
- The watermark persists in the multimedia document even after the application of common transformations.
- The watermark is difficult to remove without significant degradation of the carrying multimedia object.
- The watermark is hard to detect by an unauthorised person.
- The watermark is easy to detect by an authorised person.
- After a watermarked multimedia object is reproduced and re-captured (in the case of an image, printed and re-scanned, for example), the watermark can be extracted.