Securitymechanisms for Multimedia-Data with the Example MPEG-I-Video

Introduction

The digital Studio-TV-Signal, decribed by the CCIR Recommendation 601 has a data rate of 166 Mbit/s. None of the existing digital media has the possibility of working with this data rate and ensuring at the same time security mechanisms like 'Confidentiality' (using encryption) or 'Integrity' using (Hash-sums). The new methods, descibed by this paper, are specialy designed to ensure these security mechnisms for the high-volume video-signal of the ISO Standard 11172 (MPEG-I) **[ISO92]** for communication.

Security-mechanisms for Video-streams are needed for Multimedia-Conferences, Multimedia-Mail, professional Video-working and selling, storage or backup-purposes e.g.

This definition of the SECMPEG stream and the implementation of the SECMPEG program tries to fullfill the following requirements:

- speed
- possibility of a integration of the mechanism into the OSI Security Architecture (ISO 7498-2)
- no loss of information
- use of the DES- or RSA-function of the SECUDE package **[GMD91]**
- comparrision of the mechanisms and complete encryption

This paper is a short version of the german description of the algorythm, known as 'Sicherheitsmechanismen für Multimedia-Daten am Beispiel MPEG-I-Video'. The complete paper can be orderer at the authors and programers of the security-algorythm called 'SECMPEG':

Adresses:

Jürgen Meyer
Sonnenallee 50
12045 Berlin
GERMANY

Fon: ++ 49 30 6238622

E-mail: jm@cs.tu-berlin.de

Frank Gadegast
Leibnizstraße 30
10625 Berlin
GERMANY

Fon/Fax: ++ 49 30 3128103

E-mail: phade@cs.tu-berlin.de
phade@contrib.de

All algorythms and ideas described by the following paper are copyrigthed by the two authors.

Confidentiality

The basic idea for the confidentiality algorythm for a MPEG-I-video-stream was only to encrypt the information of the stream with the highest information contents. I-frames and intracoded macroblocks are the most valuable information parts of a MPEG-I-videostream, other informations are predicted. We tested with the manipulation of JPEG-pictures (it was not possible to manipulate a MPEG-stream and still having it decoding all right) and found that it was enough to encrypt the DC's and between 3-8 AC's to get the best results. **[MEY93.1]**

Integrity

Through long test with different, created faults in MPEG-I-Video-streams we recognised the following:

Bitfaults in MPEG-I-Video-streams are creating heavy problems for storage and usage of the streams.

- destroyed master-DC's (first DC of a frame) destroy the whole following slice
- destroyed DC's destroy the rest of the slice
- destroyed AC's destroy the rest of the block.
- bitfaults in the luminance-planes are lighten or darken the slice or block.
- bitfaults in the crominance-planes are easier to recognize, because the luminance-blocks are four times bigger than the luminance-blocks and little faults result in a total different colour, because of the two crominance-planes.
- bitfaults in I-frames result surely in more faults in the P- or B-frames (fault propagation).
- bitfaults in motion-vectors result in the moving of the contants of the frame
- missing bits (shortened stream) result in a bad reconstruction of the rest of the stream and the breakdown of the running program

So, to ensure the integrity of a MPEG-I-Video-stream, at least the following information has to be saved by a hash-sum:

- all headers and trailers
- all master-DC's and DC's of all I-frames
- all master-DC's of P- and B-frames
- and all motion-vectors

In addition the length of the stream has to be stored.

We had to choose a algorythm the ensures the following:

- it had to be very quick, because of the amount of data
- an error-correction wasnt needed nor possible
- bitfaults had to be recognized
- the additional amount of date had to be small

Results

Confidentiality

First we implmented some code based on the MPEG-Decoder of the Portable Video Reserach Group (PVRG), that decoded the MPEG-I-videostream to get our needed information and then did encryption or decryption. We found that this was not a efficient method; it needed about 30 % of the decoding time of a normal MPEG-I-videostream, the basic factor there was the Huffman-decoding with about 12-17 %. But then we realized that we could find most of the needed information without making a Huffman-decoding.

We choosed for encryption the DES (normally CBC-mode) algorythm, because it was about 100 times faster then equivalent asymetric algorythms like RSA **[MEY93.2]** and it was possible to use future hardware solutions. We used the DES-implementation out of the SECUDE package (wich is an implementation of Phil Karns DES-algorythm, see **[GMD91]** and **[KAR87]**). The next code we wrote was about 50 % of the decodingtime of a MPEG-videostream, but that was a complete encryption !

Movie	Size KB	Frame-order	Read / Write	MPEG Play	MPEG Analyser	SEC MPEG	DES Karn	DES Secude
CGS.MPG	117	IBBPBBI	0.25	19.6	6.5	12.4	9.1	20
HULA_2.MPG	148	IPPPI	0.62	72.5	23.2	33.9	13.4	28
CLAPTON.MPG	354	IBBPBBI	1.6	76.8	24.1	50.9	28.5	58
JJACKSON.MPG	447	IBBPBBI	2.4	91.5	29.8	67.2	44.3	91
TENNIS.MPG	1246	IBBPBBI	6.6	294.1	95.4	172.2	100.6	202
MEMSY2.MPG	1867	I9BP9BI	9.1	737.6	237.6	355.1	170.2	355
		KB / s	**203.1**	**3.21**	**10.05**	**6.04**	**11.37**	**5.54**
FRISCO.MPG	84	IIIII	0.3	27.4	9.6	17.1	6.9	14
MJ.MPG	619	IIIII	3	96.6	33.8	80.7	58.6	119
IICM.MPG	1679	IIIII	8.5	369.1	118.2	250.2	155.1	319
		KB / s	**201.8**	**4.83**	**14.77**	**6.84**	**10.8**	**5.26**
		KB / s	**202.6**	**3.67**	**12.41**	**6.31**	**11.08**	**5.44**

The MPEG-I-videostream consist of the following 6 Layer, whereof the first 4 layer were (at the end not suprisingly) not Huffman-coded:

- *Video-Sequence Layer:* (Random Access Unit: Context)
- *Group of Pictures Layer:* (Random Access Unit: Video Coding)
- *Picture Layer:* (Primary Coding Unit)
- *Slice Layer:* (Resynchronisation Unit)

- *Makroblock Layer:* (Motion Compensation Unit)
- *Block Layer:* (DCT Unit)

Integrity

We choosed for the integrity-mechanism the Cyclic-Redandunce-Check (CRC) using the 16- and the 32-bit-variation. It doesn't ensure a 100% certificate, but is well know, easy to implement and has quick 16-bit-variations. Other mechanisms like MD-4 or MD-5 are now prepared to get included in the SECMPEG-stream.

The speed of the implementation was the main aim of this project for integrity, therefore the decoding of the MPEG-stream to recognise all needed information was not possible, the CRC-computation was far quicker then everything else in relation to the coding or decoding of a MPEG-I-stream.

Movie	Size	Time	Time	CRC 16-bit Overhead	Ov. / MB	Time	CRC 32-bit Overhead	Ov. / MB
CG.MPG	268	0.714	1.593	0.879	3.274	2.527	1.813	6.754
MJ.MPG	619	1.538	3.681	2.143	3.460	5.879	4.341	7.009
RAIDER.MPG	978	2.418	5.769	3.352	3.425	6.813	6.962	9.231
CANYON.MPG	1744	4.286	11.648	7.363	4.222	17.912	13.626	7.813

∅			**3.595**			**7,701**

These were the first results of the not for MPEG-I-stream optimized CRC-checks. The relation between CRC-check and decoding and partial CRC-check was about 1:20. Therefore it was clear, that a complete integrity-check over the whole stream was possible and needed; a partial CRC-check was only possible, if the decoding was already done.

Therefore we only optimized the CRC-procedures for the needings of the MPEG-I-stream.

The SECMPEG-stream

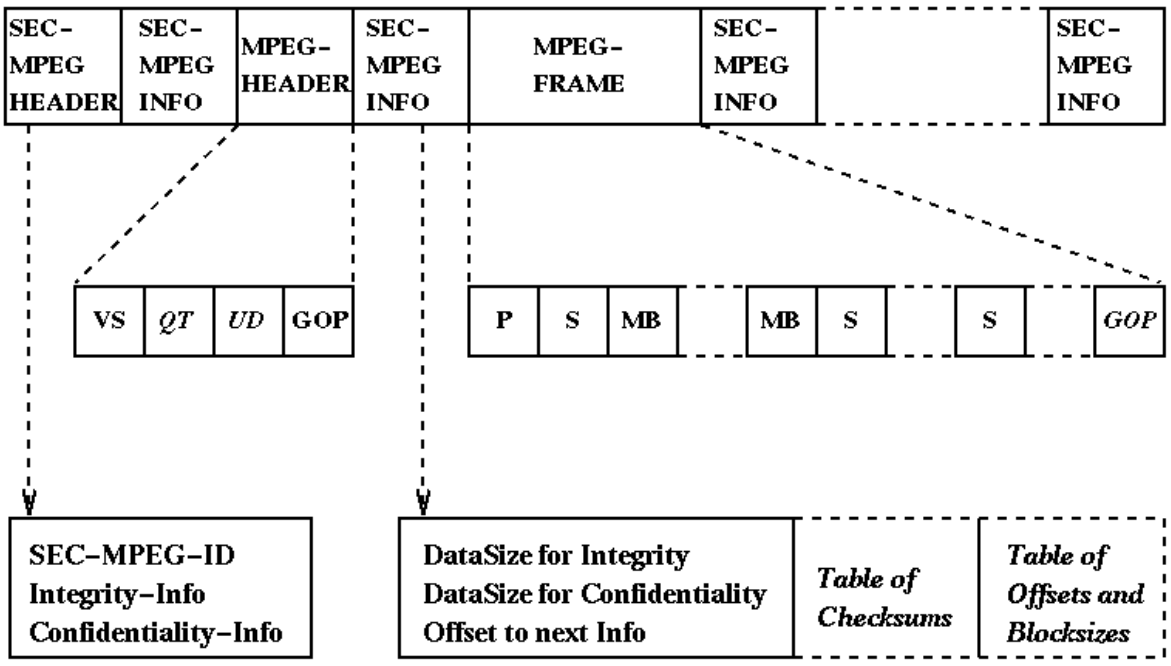
Confidentiality

We decided to include the securityinformation for the stream in front of all complete frames (the picture layer) and surely in front of the stream, because then there were no big buffers needed.

The encryption of the first 4 Layer was no problem; was a complete encryption needed, we had to extract the information of the macroblocks (how many DC's and where), so a Huffman-decoding was needed. We implemented this, but do not encrypt the so extracted DC's but the complete macroblock, because the DES is a 8 byte blockcypher and the most macroblocks were smaller than 8 bytes; a additional bytestuffing was not wanted.

The size of the additional securityinformation tables depends on the amount of intracoded macroblocks.

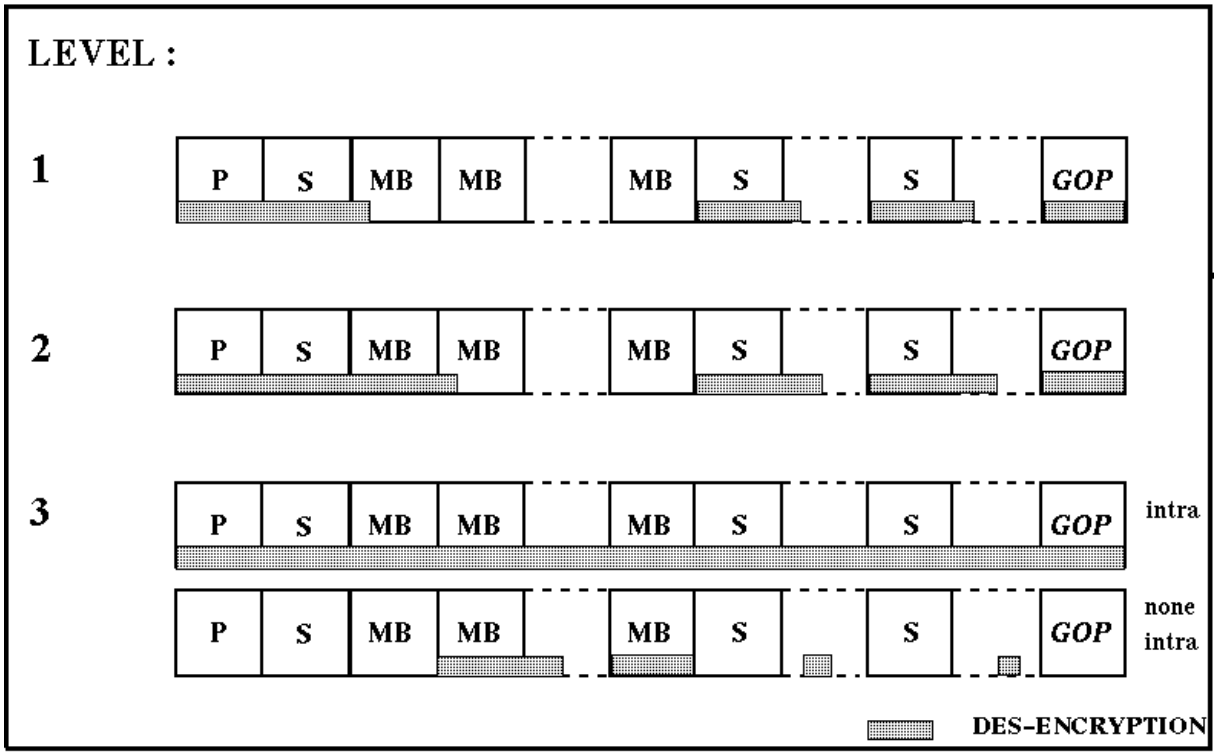
Here a diagram showing the definition of the SECMPEG-stream; a complete (but german) description is available at the authors:



Five different confidentiality-levels (C-levels) were implemented:

- **Level 0** no encryption
- **Level 1** all information off Layer 1-4 very quick, but recommended as <u>not</u> save
- **Level 2** all information off Layer 1-4 and parts of Layer 5+6 very quick and quite save
- **Level 3** all I-frames an intracoded macroblocks quick and save
- **Level 4** complete encryption

The final C-level encryption looks like this:



Integrity

Three different integrity-levels were defined:

- Playing security. All header were saved. Minimal computation time (like C-level 2).
- Origination security. All header and reference-information (like I-frame and I-frame-coded macroblocks) were saved. Still short computation time (like C-level 3).
- Stream security. All information was CRC-checked. Small computation compared to everything else (like C-level 4).

The used leved is called the I-(integrity) level.

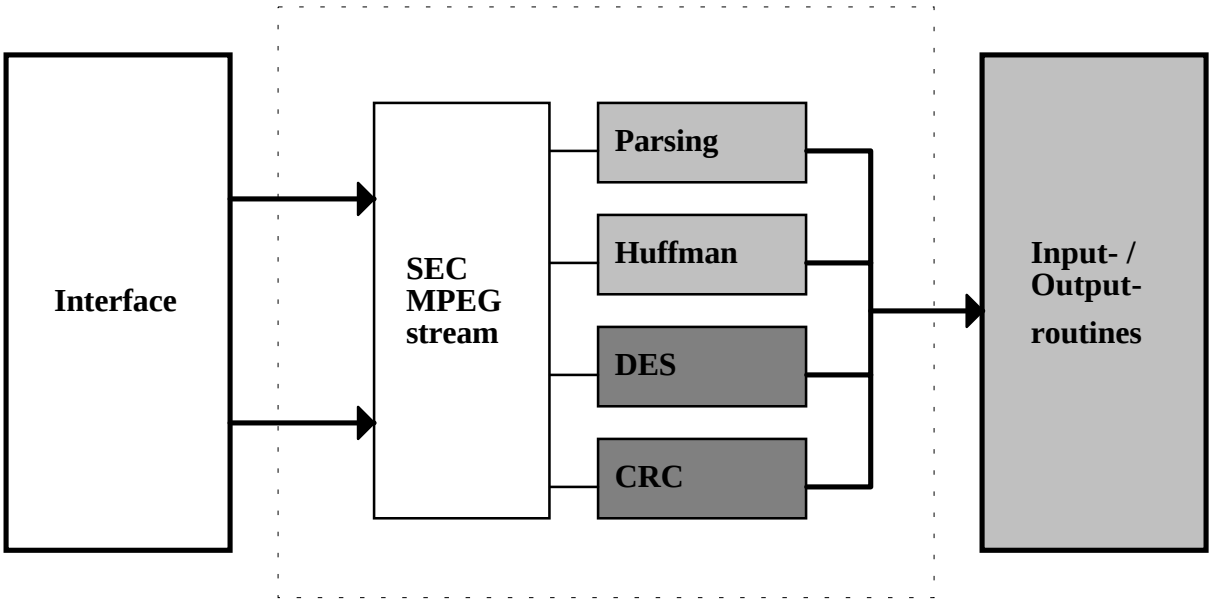
Algorythm

Three different algorythms had to be created:

- Generating of a SECMPEG-stream including the confidentiality- and/or integrity-information = encoding.
- Deleting of the security-information and rebuilding of the original stream = decoding.
- Checking of the integrity = checking.

All algorythms should work on either a file or STDIN/STDOUT.

The resulting program SECMPEG consist of the following modules (the Huffman- and Parsing-Modules base on the MPEG-coder from Any C. Hung; the DES-module bases on the DES packet from Phil Karn):



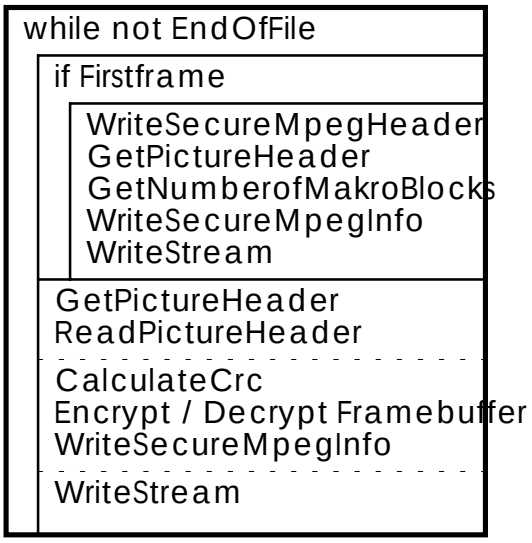
and works like follows:

- Parsing of the options
- Preparing of sevaral buffers
- Opening/creating of input/output
- Analysation of the input-stream
  - MPEG-I-stream
  - SECMPEG-I-stream (automatic recognition of C-/I-level)
- Counting of macroblocks (max.)
- Including or extraction of security-information
  - Huffman-decoding, if needed
  - Generation of Hush-sums
  - Encryption/decryption of original stream (if needed)
  - Writing of security-information
  - Writing of the (encrypted/decrypted) stream
- Termination, freeing of buffers etc.

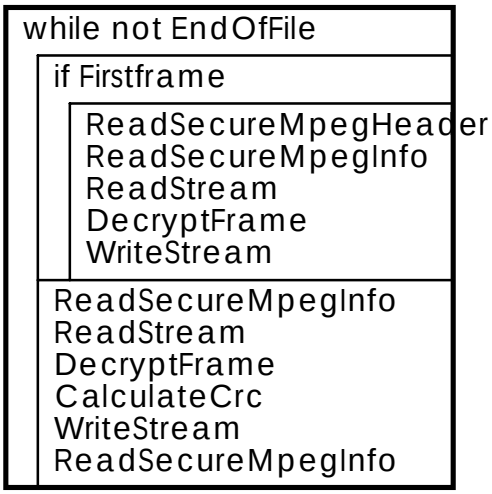
A decoding of a SECMPEG-stream includes a CRC-check.

At the end a diagram of the most important procedures of the program SECMPEG:

Prozedure EncryptMpeg ()



Prozedure DecryptSecMpeg ()



The program SECMPEG and its source code is available by the authors and works right now on:

- Sparc 10, 2+, 1 and Sun IPC, IPX using SunOS 4.x, 5.x
- i386, i486 using InterActive ISC 2.2.1, Linux, DOS

Performance

Confidentiality

The most important factor was the encryption time. We reach about 10 % of the decoding time (for Level 1) and a bit more (for Level 2) using a standard MPEG-I-stream with the framepattern IPBBBPBBBI, meaning every 0,4 seconds a referenceframe. These results were our aim. Level 3 needed much more time, but is still about 30 % (where nearly 17 % are the Huffman-decoding).

Movie	Size KB	Frame-order	MPEG Player	C-Level 1	C-Level 2	C-Level 3	CRC 16-bit	CRC 32-bit
CGS.MPG	117	IBBPBBI	19.6	4.2	5.4	17.7	1.4	1.5
HULA_2.MPG	148	IPPPI	72.5	4.2	5.1	22.7	1.8	1.9
CLAPTON.MPG	354	IBBPBBI	76.8	16.8	19.7	59.5	4.5	4.6
JJACKSON.MPG	447	IBBPBBI	91.5	23.1	24.9	86.1	6.1	6.2
TENNIS.MPG	1246	IBBPBBI	294.1	35.5	39.1	206.3	17.4	17.9
MEMSY2.MPG	1867	I9BP9BI	737.6	55.4	60.5	317.2	24.7	24.9
		KB / s	**3.21**	**30.02**	**27.01**	**5.89**	**74.75**	**73.31**
FRISCO.MPG	84	IIIII	27.4	2.8	3.1	11.5	1	1
MJ.MPG	619	IIIII	96.6	14.7	20.8	86.2	8.3	8.4
IICM.MPG	1679	IIIII	369.1	56.3	82.7	238.7	23.9	24.4
		KB / s	**4.83**	**32.27**	**22.34**	**7.08**	**71.74**	**70.47**
		KB / s	**3.67**	**30.80**	**25.01**	**6.22**	**73.63**	**72.25**

Table 1: SEC-MPEG Performance (AT-386 / 40MHz / InterActive Unix 2.2.1)

Decryption is always about a factor 2 quicker, because there is no analysation to be done.

We know, that a attack on the so encrypted streams is possible, but attacking a Level 1 or 2 SECMPEG-stream means to compute the starting picture with bruteforce-methods. First some offsets have to be found, then some not encrypted data, like the intra-coded macroblocks can be reconstructed. If this is done, the rest of the stream can be reconstructed easily. To attack a Level 3 SECMPEG-stream is nearly impossible, it is as complicated as breaking the DES.

Integrity

The results of the first optimized implementations of the 16- and 32-bit-CRC's were impressive:

Movie	Size	Time	Time	CRC 16-bit Overhead	Ov. / MB
CG.MPG	268	0.714	1.593	0.879	3.274
MJ.MPG	619	1.538	3.681	2.143	3.460
RAIDER.MPG	978	2.418	5.769	3.352	3.425
CANYON.MPG	1744	4.286	11.648	7.363	4.222
		∅			**3.595**

Table 2: CRC - 16 Bit (AT-386 / 40 Mhz / DOS 5.0)

Movie	Size	Time	Time	CRC 32-bit Overhead	Ov. / MB
CG.MPG	268	0.714	2.527	1.813	6.754
MJ.MPG	619	1.538	5.879	4.341	7.009
RAIDER.MPG	978	2.418	6.813	6.962	9.231
CANYON.MPG	1744	4.286	17.912	13.626	7.813
		∅			**7,701**

Table 3: CRC - 32 Bit (AT-386 / 40 Mhz / DOS 5.0)

A complete integrity-check was possible. The high overhead with the 32-bit-CRC depended on the first test platform: a DOS machine; the 32-bit-instruction were emulated with 16 bit code. The code was nearly same as quick as the 16-bit-results on a real 32-bit-system (see Table 1). So a integrity-check for parts of the stream is only useful when the needed information is already decoded (because of a additional encryption).

Future

Right now we are working (W) and thinking (T) of the following enhancements of SECMPEG:

- additional encryption by using a special quick stream-cypher (W)
- checking of other encryption standards (T)
- including MD-4 and MD-5 for integrity (W)
- checking if RSA is useful (W)
- porting of SECMPEG for use with MPEG-I-audio and MPEG-I-system (T)
- porting of SECMPEG for use with MPEG-II (T)
- porting of these methods to other multimedia-streams (DVI, AVI) (T)
- parrallel computing (T)
- mathematical checks if the used algorythms are really save (W)

It will be welcome to integrated these methods into the the known MPEG-I-video encoder and decoder.

Appendix A: Further inormation

[CHI92] Chiariglione, Leonardo: Multimedia Communication (MPEG-II), Brussels 1992

[CT93] c't (Kürzel ku): MPEG-2-Standard festgeklopft, Heft 6, 1993

[GAD93] Gadegast, Frank: Offene Dokumentverarbeitung mit multimedialen Standards, Referat, Jan. 1993

[GAD93.2] Metin Çetinkaya, Gadegast, Frank: MPEG-Standard ISO 1172, Referat, Jul. 1993

[GAD93.3] Metin Çetinkaya, Gadegast, Frank: Graphische Benutzeroberflächen und die Multimediafähigkeit, Referat, Jun. 1993

[GMD91] Wolfgang Schneider: SecuDE, Gesellschaft für Mathematik und Datenverarbeitung, 1991

[HUN93] Hung, Andy C.: PVRG-MPEG-CODEC 1.1, Manual, Stanford, 1993

[HUN93.2] Hung, Andy C.: PVRG-JPEG-CODEC 1.1, Manual, Stanford, 1993

[ISO90.4] ISO/IEC International Standard : Standard Music Description Language (SMDL), Hypermedia/Time-base Subset (HyTime), International Organization for Standardization, Geneva, 1990

[ISO92] ISO/IEC Draft International Standard (DIS) 11172: Information technology - Coding of moving pictures and associated audio for digital storage media up to about 1,5 Mbit/s (MPEG), International Organization for Standardization, Geneva, 1992

[KAR87] Phil Karn : The DES package, 1987

[MEY93.1] Meyer Jürgen : Vertraulichkeit von Multimedia-Daten , Konzeptpapier, Technische Universität Berlin, Mai / Juni 1993

[MEY93.2] Meyer Jürgen : CRYPT, Datenverschlüsselung mit Secude, Technische Universität Berlin, Mai 1993

[MUS93] Musman, Hans-Georg; Werner, Oliver; Fuchs, Hendrik: Kompressionsalgorythmen für interaktive Multimedia-Systeme, IT+TI 2/93, Hannover 1993

[PAT92] Patel, Ketan; Smith, Brian C.; Rowe, Lawrence A.: Performance of a Software MPEG Video Decoder, Berkeley, 1992

Ask Frank Gadegast (phade@cs.tu-berlin.de) for the MPEG-Frequently-Asked-Question-File to be up to date with the revolution of MPEG-I.