

Projektbericht:

# Sicherheitsmechanismen für Multimedia- Daten am Beispiel MPEG-I-Video

zum Projekt: Sicherheit in verteilten Systemen

SS 93, LV 2032 L 470  
TU Berlin

von Jürgen Meyer und Frank Gadegast

**Adressen:** Jürgen Meyer  
Sonnenallee 50  
12045 Berlin  
GERMANY

Frank Gadegast  
Leibnizstraße 30  
10625 Berlin  
GERMANY

**Fon:** ++ 49 30 314-25157

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

**Fon/Fax:** ++ 49 30 3128103

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

# Inhaltsverzeichnis:

<b>1. Aufgabenstellung</b> .....	<b>1</b>
<b>2. Zusammenfassung des Konzeptpapiers</b> .....	<b>1</b>
<b>2.1. Integrität</b> .....	<b>1</b>
<b>2.2. Vertraulichkeit</b> .....	<b>2</b>
<b>2.3. Vorgehensweise</b> .....	<b>2</b>
<b>3. Überprüfung der Umsetzbarkeit des Konzeptes</b> .....	<b>2</b>
<b>3.1 MPEG-Analyser</b> .....	<b>2</b>
<b>3.2. DES</b> .....	<b>3</b>
<b>3.3 Der MPEG-Video-Stream</b> .....	<b>3</b>
<b>3.4. Abweichungen vom Konzept</b> .....	<b>6</b>
<b>4. Implementation</b> .....	<b>6</b>
<b>4.1. SECURE-MPEG-STREAM</b> .....	<b>6</b>
<b>4.2. Integrität</b> .....	<b>8</b>
<b>4.3. Vertraulichkeit</b> .....	<b>8</b>
<b>4.4. Ablaufbeschreibung</b> .....	<b>9</b>
<b>5. Ergebnisse</b> .....	<b>11</b>
<b>5.1 Integrität</b> .....	<b>11</b>
<b>5.2. Vertraulichkeit</b> .....	<b>12</b>
<b>5.3. Einschätzung der Sicherheit des Mechanismus</b> .....	<b>13</b>
<b>6. Ausblick</b> .....	<b>15</b>
<b>Literaturangaben</b> .....	<b>16</b>

# 1. Aufgabenstellung

Im Rahmen des Projektes **Sicherheit in verteilten Systemen** hatten wir die Aufgabe Sicherheitsmechanismen für Multimedia-Daten zu entwerfen. Beispielhaft für Multimedia-Daten soll **MPEG-I**, ein Format für Bewegtbilder, untersucht werden.[**ISO92**]

Der Mechanismus soll später in Sicherheitsdienste integriert werden, die in der **OSI Security Architecture** (ISO 7498-2) spezifiziert sind. Weiterer Rahmenbedingen sind:

- kein größerer Informationsverlust durch den Einsatz der Sicherheitsmechanismen
- Einsatz der von **Secude** zur Verfügung gestellten RSA- und DES-Funktionen [**GMD91**]
- Effizienzvergleich zwischen dem entwickelten Mechanismus und einer vollständigen Verschlüsselung

## 2. Zusammenfassung des Konzeptpapiers

### 2.1. Integrität

Um die Integrität der Daten zu gewährleisten, wurde zuerst untersucht, welche Auswirkungen Veränderungen im MPEG-I-Datenstrom (egal ob verursacht durch einen Angreifer oder durch Übertragungs- und Speicherfehler) haben. Je nachdem wo in dem MPEG-Video-Stream Fehler bei der Übertragung bzw. Speicherung auftreten, sind auch die zu beobachtenden Fehler des Videobildes unterschiedlich. Da es sich bei MPEG um ein hochkomprimierendes Datenformat (fast redundanzfrei) handelt, sind Fortpflanzungsfehler (Fehlerpropagierung) nicht zu vermeiden und bedürfen einer besonderen Analyse. Schlußfolgerung ist, daß ein Integritäts-Check bezogen auf die Länge des Video-Streams bei Speicherung unabdingbar wird, d.h. das insbesondere Layerinformationen geschützt werden müssen.[**GAD93.4**]

Zusammenfassend sind folgende Daten des MPEG-Streams zu schützen:

- der gesamte Header
- alle DC's (MDC's sowie DDC's) beider Farb-Planes aller I-Frames
- alle MDC's der Grauwert-Plane aller I-Frames, sowie alle anderen MDC's in B- oder P-Frames, egal ob Grauwert-Plane oder Farb-Plane
- alle Motion-Vektoren
- der gesamte Trailer (falls vorhanden)
- Zusätzlich ist eine Check-Summe betreffs der Anzahl der Bytes jedes Frames zu erstellen, um eine Verkürzung des MPEG-Streams überwachen zu können. Verkürzte Frames sind beim dekodieren eventuell zu entfernen.

Es ist ein Verfahren zum Integritätsschutz zu wählen, das mindestens folgende Anforderungen erfüllt:

1. Im Hinsicht auf die aufwendige Kodierungen des MPEG-I-Stream muss das gewählte Verfahren extrem schnell sein.
2. Bitkipper oder Manipulationen am MPEG-I-Stream müssen erkannt werden. Eine mögliche Fehlerkorrektur kann in Hinsicht auf die hochvolumigen Daten nicht zu Verfügung gestellt werden, da sie immer mit dem Einfügen von redundaten Daten verbunden ist.

3. Die Anzahl der zusätzlich in den Stream einzufügenden Daten (Steuerdaten) muss so gering wie möglich gehalten werden.

## 2.2. Vertraulichkeit

Die Grundidee für die Verschlüsselung des MPEG-Streams war, nur die Headerinformation und die Daten mit dem höchsten Informationsgehalt zu schützen. Für die Wiederherstellung der komprimierten Videosequenz sind die Intra-Frames und die intra kodierten Makroblöcke von besonderer Wichtigkeit. Um nun nicht die kompletten Datenblöcke zu verschlüsseln wurde untersucht, in welchem Umfang DC- und AC-Koeffizienten zu sichern sind. Das Ergebnis war, daß eine Anzahl von 3 - 8 AC-Koeffizienten und des DC-Koeffizienten ausreichend sein sollte, um eine hinreichend gute Verschlüsselung zu gewährleisten. Als Verschlüsselungsmechanismus soll der DES angewendet werden.[MEY93].

## 2.3. Vorgehensweise

Um die zur Verschlüsselung relevanten Daten aus dem MPEG-Stream zu extrahieren, ist es notwendig, den Stream zu analysieren, dies setzt allerdings eine Huffman-Dekodierung voraus. Eine Aussage über die Performance der **Analyse der Layerebene** findet man in [PAT92], sie beträgt dort 17,4 % der Gesamtzeit für das Abspielen eines MPEG-Videos. Wir waren der Meinung, diesen Wert noch unterbieten zu können, da wir nicht alle Informationen benötigen, die diesem Wert zugrunde liegen. Bei dieser Vorgehensweise erhofften wir uns einen Wert von < 30 % , inklusive der DES-Verschlüsselung. Aus Performancegründen wollten wir aber versuchen, die Analyse und Verschlüsselung direkt auf dem **Bitstream** vorzunehmen.

# 3. Überprüfung der Umsetzbarkeit des Konzeptes

## 3.1 MPEG-Analyser

In Anlehnung an den Code der Portable Video Research Group (PVRG) haben wir einen MPEG - Parser- und Dekoder implementiert.[HUN93]. Dieser Parser und Dekoder war ausgelegt, die Stellen des MPEG-Streams zu extrahieren, die wir für eine Verschlüsselung vorgesehen hatten, d.h. eine Rekonstruktion der Makroblöcke wurde nicht vorgenommen. Bei den folgenden Untersuchungen mussten wir leider feststellen, daß wir nur auf einen durchschnittlichen Wert von ~30 % der Zeit, die für das Abspielen benötigt wird, gekommen sind und nicht auf einen Wert unter 17 %, wie wir angenommen hatten. Dabei hatten wir schon an einigen Stellen im Code Verbesserungen vorgenommen, die sich positiv auf das Laufzeitverhalten auswirkten. Im Laufe des Projekts wurde uns ein weiterer MPEG-Analyser [MFILT] zur Verfügung gestellt. Dieser lag mit seinen Performancewerten noch deutlich über den Werten unserer Implementierung, wobei aber zu berücksichtigen ist, daß hier noch weitere Informationen ausgewertet werden. Dies hat im Nachhinein unsere Entscheidung gerechtfertigt, nicht noch durch weiteres Tuning des Algorithmus diesen Wert anzustreben, sondern das Konzept in Teilen zu überarbeiten. Die Performance-Untersuchungen des Berkely-Players sind unserer Meinung als Referenz zu hinterfragen.

### 3.2. DES

Für die Verschlüsselung haben wir den **DES** vorgesehen, zum einen ist der Algorithmus erheblich effizienter als asymmetrische Verfahren (Faktor 100 gegenüber RSA) und zum anderen besteht eventuell die Möglichkeit, in Zukunft Hardwarelösungen einzusetzen. Die DES-Funktionalität sollte dem **Secude-Paket** entnommen werden.[GMD91] Wir haben dann die Zeitdauer einer vollständigen Verschlüsselung mit dem DES und der benötigten Zeit für das Abspielen der Videosequenz verglichen und festgestellt, daß eine vollständige Verschlüsselung nur geringfügig schneller ist als das Abspielen. Daraufhin haben wir das DES - Paket von **Phil Karn** mit einem geeigneten Interface ausgestattet und getestet.[KAR87] Die Ergebnisse die damit erzielt wurden, waren um den Faktor 2 schneller, d.h. eine vollständige Ver- und Entschlüsselung benötigt weniger Zeit als das Abspielen. Weiterhin haben wir festgestellt, das Secude auch auf das Software-Paket von Karn aufbaut. Bei einer späteren Integration des Mechanismus in eine Sicherheitsplattform, die Secude benutzt, ist also nur eine geringfügige Änderung der Modulschnittstelle nötig.

Movie	Size KB	Frame-order	Read / Write	MPEG Play	MPEG Analyser	MFILT	DES Karn	DES Secude	
CGS.MPG	117	IBBPBBI	0.25	19.6	6.5	12.4	9.1	20	
HULA_2.MPG	148	IPPI	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	
			<b>KB / s</b>	<b>203.1</b>	<b>3.21</b>	<b>10.05</b>	<b>6.04</b>	<b>11.37</b>	<b>5.54</b>
FRISCO.MPG	84	IIII	0.3	27.4	9.6	17.1	6.9	14	
MJ.MPG	619	IIII	3	96.6	33.8	80.7	58.6	119	
IICM.MPG	1679	IIII	8.5	369.1	118.2	250.2	155.1	319	
			<b>KB / s</b>	<b>201.8</b>	<b>4.83</b>	<b>14.77</b>	<b>6.84</b>	<b>10.8</b>	<b>5.26</b>
			<b>KB / s</b>	<b>202.6</b>	<b>3.67</b>	<b>12.41</b>	<b>6.31</b>	<b>11.08</b>	<b>5.44</b>

**Tabelle 1: Performance von MPEG-Analyser und vollständiger DES-Verschlüsselung**

### 3.3 Der MPEG-Video-Stream

Ein MPEG-Video-Stream wird durch Aneinanderreihung von Intra- (I), Predicted- (P), und Bidirectional- (B) beschrieben. Jeder Frame wird wiederum in drei Planes, eine Luminanz-Plane (Grauwerte, Y) und zwei Chrominanz-Planes (Farbwerte, Cr und Cb), zerlegt. Alle Planes werden in sogenannte Macroblöcke unterteilt. Die Y-Plane wird in 8x8 Pixel-Blöcke, die Cr-/Cb-Planes in 16x16 Pixel-Blöcke aufgeteilt. Jeder dieser Blöcke wird mittels einer Discrete-Cosine-Transform (DCT) kodiert, dabei wird der erste Wert jedes Blocks als DC-Koeffizient (DC) bezeichnet; die restlichen Differenzwerte als AC-Koeffizienten (AC). DC's

werden untereinander auch als Differenzen gespeichert. "Echte" DC's bezeichnen wir als Master-DC's (MDC); einen als Differenz zu einem MDC gespeicherten DC als Differenz-DC (DDC). Zusätzlich können Frames auch als Verweise der Macroblöcke auf gleiche, in vorherigen Frames gespeicherten Blöcken kodiert werden. Dieses Verfahren nennt man Motion Compensation.

Der MPEG-I-Standard sieht eine Layerstruktur vor, durch die auch gleichzeitig die Syntax des Bitstreams eines MPEG-kodierten Videos charakterisiert wird. Es existieren im ganzen 6 Layer, die jeweils spezielle logische und Signalprozessing-Funktionen zur Verfügung stellt.

Layer 1 - 6 :

- *Video-Sequence Layer:* (Random Access Unit: Context)  
Allgemeine Parameter des Videostreams
- *Group of Pictures Layer:* (Random Access Unit: Video Coding)  
Stellt Zugriffspunkte jeweils an den Intraframes zur Verfügung
- *Picture Layer:* (Primary Coding Unit)  
Zugriff auf einzelne Bilder
- *Slice Layer:* (Resynchronisation Unit)  
Wiederaufsetzpunkte innerhalb eines Frames
  
- *Makroblock Layer:* (Motion Compensation Unit)
- *Block Layer:* (DCT Unit)

Während der Arbeit am MPEG-Analyser mussten wir eine Annahme über den Aufbau des Streams korrigieren, die wir bei der Ausarbeitung des Konzepts zugrunde gelegt hatten. Die oberen 4 Layer des MPEG-Video-Streams, die allesamt logische Funktionen zur Verfügung stellen, werden nicht mittels einer Huffman-Entropie-Kodierung kodiert.

Weiterhin kann allen oben genannten Header eine weiterer Header folgen. Dieser Header, Extended User Data, ist für Erweiterungen reserviert, wird aber von manchen Herstellern von MPEG-Encodern schon benutzt.

Allen Headern ist ein Marker vorangestellt, dem ein zusätzliches Byte zur Identifizierung des Headers folgt. Weiterhin sind die Marker bytealigned **[ISO92]**

Marker : 0x00,0x00,0x01 Trailer : 0x00 - 0xFF

Der Group-Of-Pictures und Picture-Layer sind Aufsetzpunkte für die Videokodierung. Der Group-Of-Pictures-Header beinhaltet die Information über die Zusammensetzung des Videostreams (Anzahl und Abfolge der Intra-, Predicted-, Bidirectional- und DC Intra-Frames). Der Picture-Layer wird benötigt, um Aufsetzpunkte für Vor- und Rückspielung usw. zu Verfügung zu stellen. Der Slice-Header ist in erste Linie für die Synchronisation und Resynchronisation im Fehlerfall zuständig. Die Anzahl der Slice-Header ist in der Spezifikation nicht vorgegeben und richtet sich in erster Linie nach dem vorgesehen Einsatz des Videos, sowie der Implementierung des Encoders.

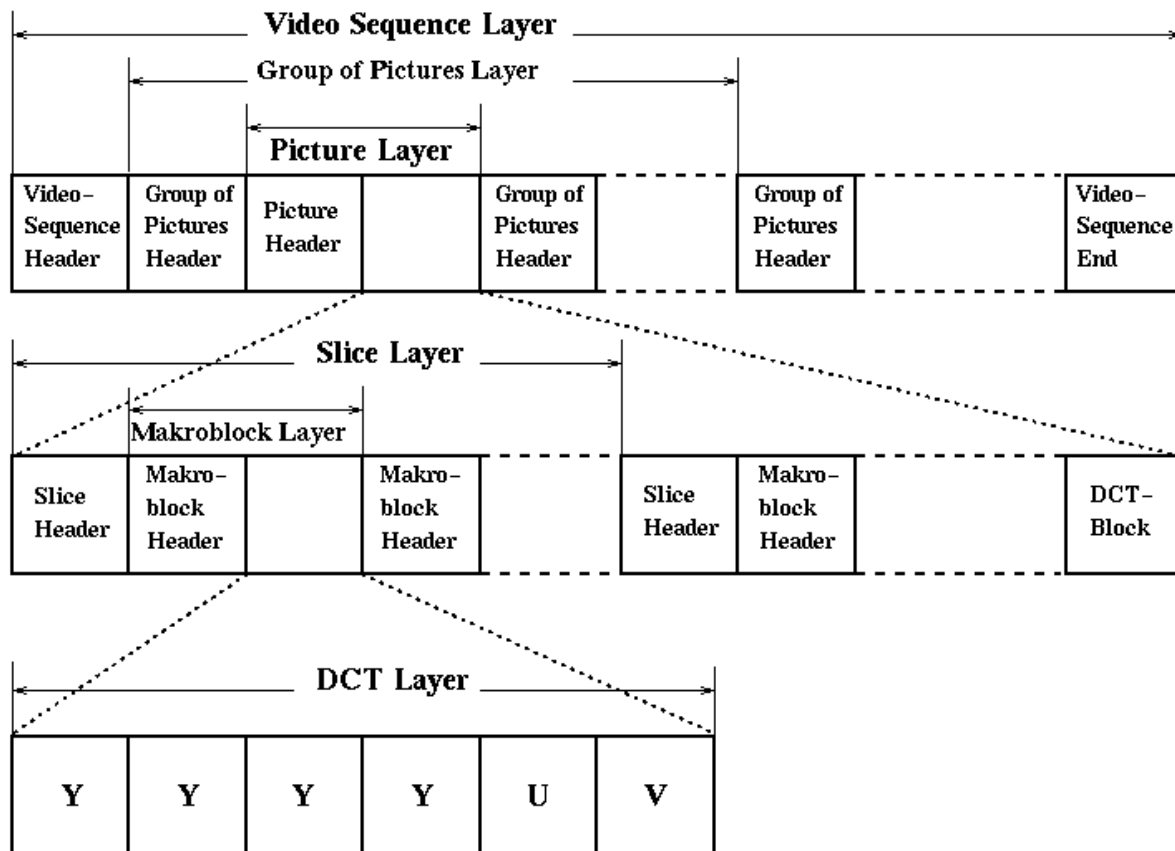


Abbildung 1: MPEG-Video-Stream

### 3.4. Abweichungen vom Konzept

Aufgrund der gewonnenen Erkenntnisse über den Aufbau des MPEG-Streams im Detail und die mangelhafte Performance des Dekodiervorgangs haben wir das ursprüngliche Konzept in Teilen modifiziert. Die Verschlüsselung der Headerinformation gestaltet sich für die oberen 4 Layer einfacher als erwartet, dagegen ist die Verschlüsselung der relevanten Daten im Makroblock- und DCT-Layer nur bedingt sinnvoll zu realisieren. Ist ein direkter Zugriff auf einzelne Makroblock-Header erwünscht, ist eine vollständige Dekodierung des Streams zwischen zwei Headern der Layer 1-4 notwendig. Diese Möglichkeit haben wir noch implementiert, aber von der gezielten Verschlüsselung von DC- und AC-Koeffizienten auf der DCT-Ebene haben wir Abstand genommen. Liegt ein MPEG-I-Stream vor, der der Spezifikation entspricht (z.B.: 352 x 240, jeder 12. Frame ein Intraframe ~ 20kByte), hat jeder DCT-Block eines intra-kodierten Frames eine durchschnittliche Größe von max. 10 Byte. Der DES ist ein Blockcipher mit einer Blockgröße von 8 Byte. Bei der Verschlüsselung nur der relevanten Daten eines DCT-Blocks, wäre eine Zusammenfassung der Daten aus mehreren DCT-Blöcken notwendig, um auf eine Blockgröße von 8 Byte zu kommen. Bei MPEG-I-

Streams haben wir deshalb diese Möglichkeit verworfen, da der entstehende Mehraufwand bei der Kodierung höher ist, als der Aufwand für das Verschlüsseln des ganzen Makroblockes. Wir Verschlüsseln aus diesen Gründen bei Predicted- und Bidirektionalen-Frames, je nach Qualität des Services, die intra-kodierten Makroblöcke vollständig. In wie weit diese Ergebnisse auch auf MPEG-II zu übertragen sind, oder ob sich neue Ansätze ergeben, kann erst nach Studium des Drafts beurteilt werden.

## 4. Implementation

### 4.1. SECURE-MPEG-STREAM

Die Vorgabe bei der Entwicklung des Mechanismus ist eine spätere Integration in Security-Dienste gemäß der OSI-Security-Frameworks, speziell an einen Einsatz in Message-Handling-Systemen (MHS / CCITT X.400). Da eine Spezifikation für eine Sicherheitsplattform, die die Integration der verschiedenen Dienste beinhaltet, noch aussteht, kann noch nicht mit Sicherheit gesagt werden auf welcher Basis wir aufsetzen können. Laut Vorgabe ist zuerst nur an die Verarbeitung vorhandener MPEG-Streams gedacht, da es sich hier um hochvolumige Daten

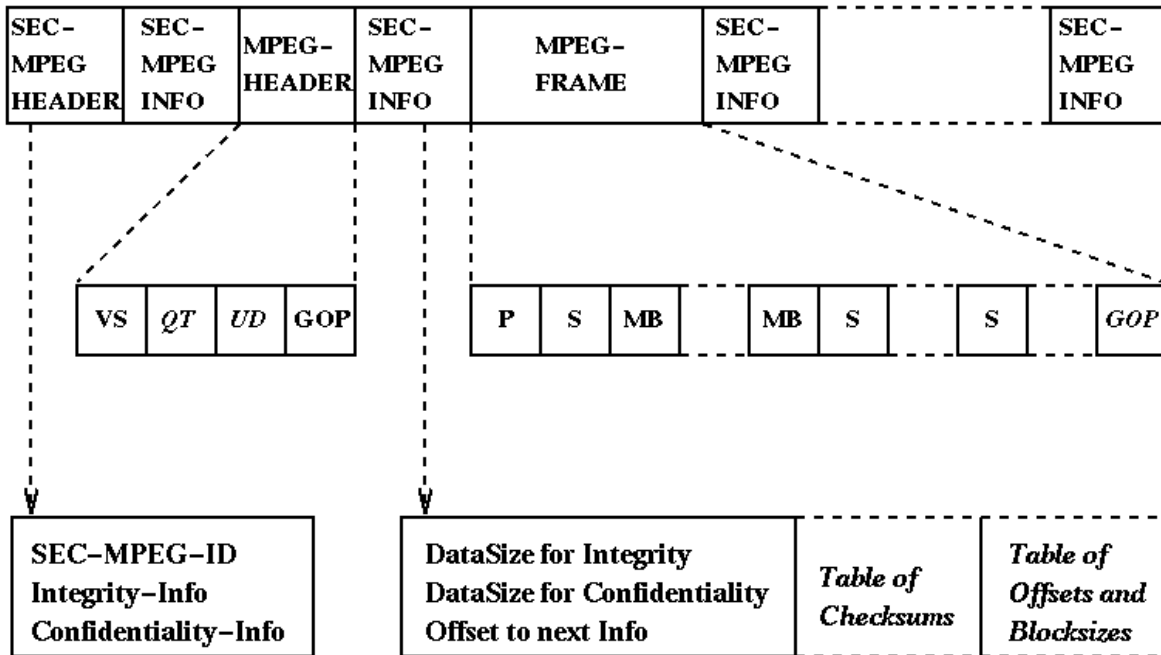


Abbildung 2: Sec-Mpeg-Stream

handelt, haben wir uns für einen stream-orientierten Ansatz entschlossen. Weitere Prämissen beim Design des Algorithmus waren die Hardwareunabhängigkeit und eine schnelle Dekodierung des geschützten Streams.

Der MPEG-Stream wird vor der eigentlichen kryptologischen Verarbeitung in Blöcke zerlegt. Es bieten sich hier nur zwei Möglichkeiten an, zum einen eine Aufteilung in Blöcke des Group-Of-Pictures-Layer oder in Blöcke des Picture-Layers, andere Aufteilungen sind nicht sinnvoll. Wir haben uns für die zweite Variante entschieden, da zum einen der Speicherbedarf einen definierten Wert nicht übersteigt und somit auf verschiedenen Rechnerarchitekturen implementiert werden kann, andererseits dies einer möglichen parallelen Verarbeitung entgegenkommt. Dem eigentlichen Stream ist in der Abbildung ein Header (SEC-Header) vorangestellt. Prinzipiell ist dieser Header überflüssig, da er nur Informationen beinhaltet, die in der Regel Bestandteil des Übertragungsprotokolls sind. Da, wie oben schon erwähnt, noch Unklarheiten bestehen, wir aber eine Möglichkeit zum Testen des Algorithmus benötigen und auch den Mechanismus interessierten Benutzern zur Verfügung stellen möchten, haben wir diesen Header definiert. Um die Performance des Dekodiervorgangs zu verbessern wird der MPEG-Stream nur beim Kodierdurchgang analysiert und die notwendigen Informationen in einer Tabelle gespeichert. Vor der Weiterverarbeitung besteht diese aus den Offsets der Headern der Layer 1-5. In einem weiteren Bearbeitungsschritt wird die Länge der zu verschlüsselnden Blöcke bestimmt und den Einträgen zugeordnet. Anschließend wird die Konsistenz der Tabelle überprüft. Da die Länge dieser Information variabel ist, wird jedem Block ein SEC-Info zugeordnet, indem der Umfang der folgenden Daten und der Offset zum nächsten SEC-Info vermerkt ist. Diese zusätzlichen Daten werden anschließend verschlüsselt.

Der Umfang der Tabellen ist Abhängig von der Anzahl der Makroblöcke. Die maximale Größe einer Tabelle beträgt 1320 Byte bei einer Bildfrequenz von 30 Hz. ( max 330 Makroblöcke x Größe eines Eintrags) Bei den Testdurchläufen haben wir festgestellt, daß im Durchschnitt nur 20% - 30% der Makroblöcke in den Predicted- und Interpolated-Frames aus intra-kodierten Blöcken besteht. Der mittlere Größenzuwachs des SEC-MPEG-Streams ist gegenüber dem Effizienzgewinn beim Dekodiervorgang zu vertreten.

## **4.2. Integrität**

Drei Abstufungen im Integritätsschutz wurden definiert und implementiert:

- Abspielschutz. Alle Header werden geschützt. Minimaler Rechenaufwand.
- Originalitätsschutz. Alle Header, sowie alle Referenzbilder (I-Frames und intracodierten Macroblöcke) werden geschützt. Der Rechenaufwand ist noch gering.
- Stream-Schutz. Der komplette Stream wird mit einer Hash-Summe geschützt.

Den jeweiligen Schutzmechanismus bezeichnen wir als I-(Integritäts) Level. Die Hash-Summen werden immer über dem Originalstream erzeugt.

## **4.3. Vertraulichkeit**

Fünf Abstufungen im Vertraulichkeitsschutz wurden definiert und implementiert:

Level	Verschlüsselung
Level 0	keine
Level 1	alle Informationen der Layer 1 - 4
Level 2	alle Informationen der Layer 1 - 4 und zum Teil der Layer 5 & 6
Level 3	Intra-Frames vollständig, sonst alle intra-kodierten Makroblöcke
Level 4	vollständig

Den jeweiligen Schutzmechanismus bezeichnen wir als C-(Confidentiality) Level. Die Mechanismen der Ebenen 1 - 3 setzen sich bei der Verschlüsselung aus einer Untermenge der 4 Basisoperationen zusammen:

- Parsen des Streams und Erstellen einer Tabelle je Frame.
- Dekodierung der Huffman kodierten Layer und Erstellung einer Tabelle.
- Berechnung der Blockgröße für die Verschlüsselung und Konsistenzprüfung.
- Verschlüsselung eines Blockes mit dem DES (Wahl des Modus nach der Blockgröße)

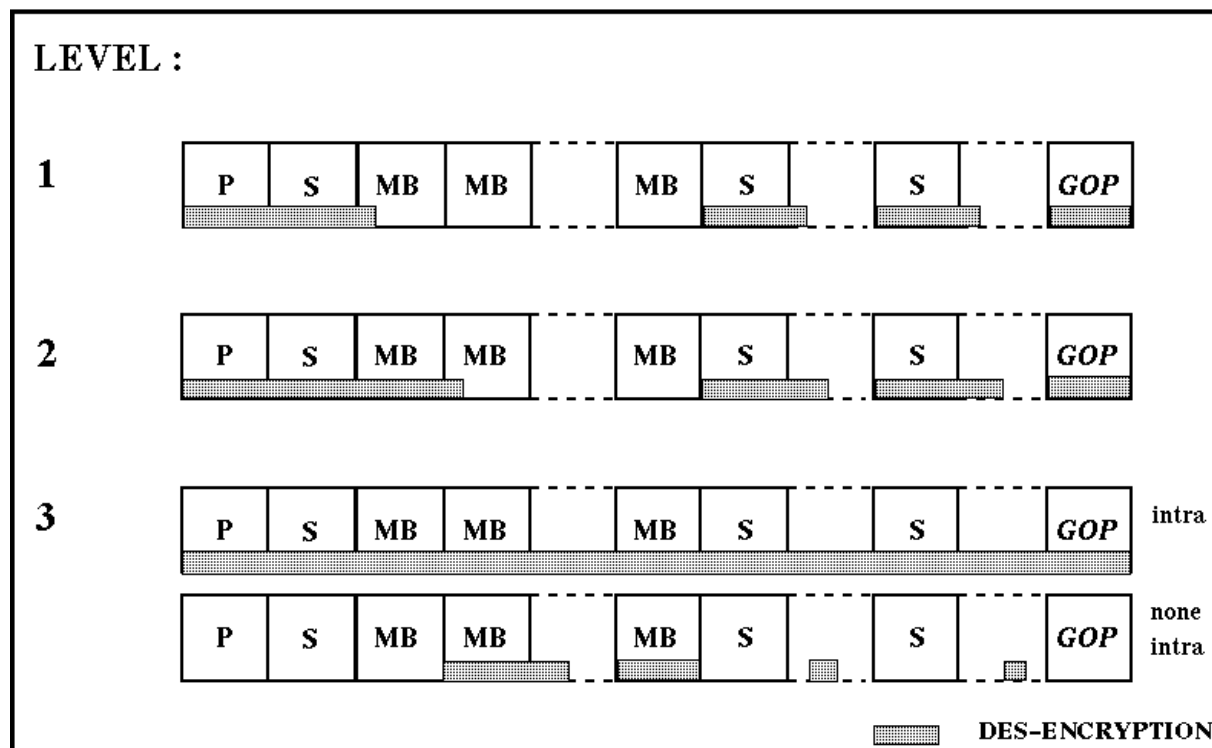


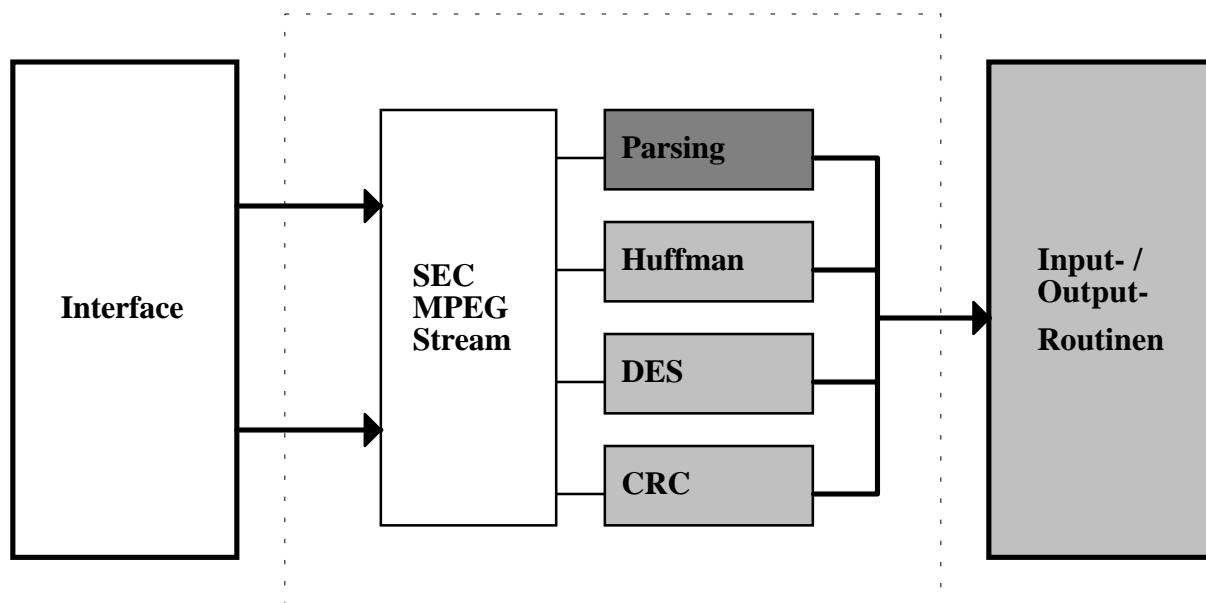
Abbildung 3: Security-Level Confidentiality

## 4.4. Ablaufbeschreibung

Mehrere Modi müssen bei der Verarbeitung eines SECMPEG-Stream unterschieden werden:

- Generierung eines SECMPEG-Stream mit Vertraulichkeits- und/oder Integritätsinformationen = 'Encoding'.
- Entfernungen dieser Steuerinformationen (mit Integritätscheck) = 'Decoding'.
- Überprüfung der Integrität = 'Checking'.

Alle Modi können entweder auf ein File angewendet werden, oder über die Schnittstellen STDIN/STDOUT ein- und ausgegeben werden. Ein Integritätscheck erzeugt keinen Ausgabestream.



**Abbildung 4: Modulstruktur von SECMPEG**

Die Module Huffman und Parsing basieren zu großen Teilen auf der Implementierung des MPEG-Codec's von Andy C. Hung[HUN93].

Das Modul DES ist dem DES-Packet von Phil Karn entnommen und um ein Interface erweitert worden[KAR87].

Die hellgrau hinterlegten Module sind austauschbar.

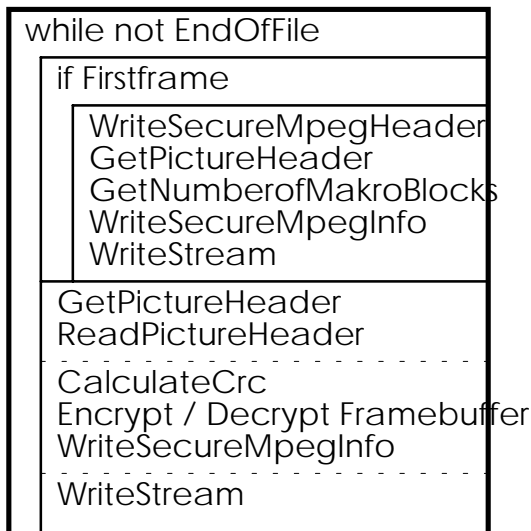
Der Ablauf des Programms SEC-MPEG stellt sich wie folgt dar:

- Parsen der Kommandozeile
- Festlegen von Buffern
- Öffnen/Erzeugen der Ein-/Ausgabestreams
- Analyse des Eingabestreams
  - MPEG-Stream
  - SEC-MPEG-Stream (automatische Erkennung des Streams und der C-/I-Level)
- Anzahl Macroblöcke
- Einfügen/Herausnehmen der Steuerinformationen entsprechend der C- und I-Level
  - Huffman-Dekodierung des Streams (falsch gewünscht)
  - Generierung der Hash-Summen
  - Verschlüsselung/Entschlüsselung des Stream
  - Schreiben der Steuerinformationen
  - Schreiben des (verschlüsselten) Streams
- Termination, Buffer freigeben etc.

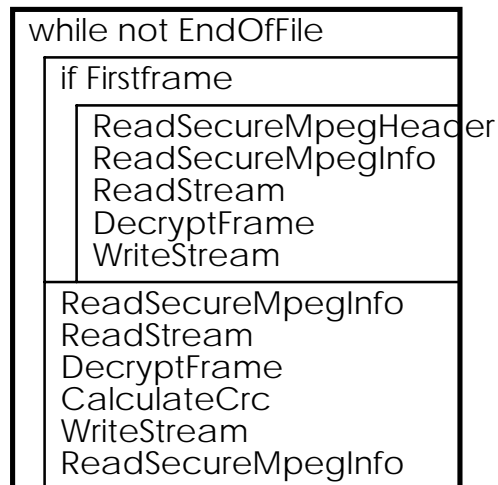
Bei der Dekodierung eines SEC-MPEG-Streams werden nicht nur die Steuerinformationen entfernt und der Stream encrypted (falls gecrypted), sondern auch die Hash-Summe überprüft (falls mit CRC kodiert).

Schlußendlich sollen hier noch die beiden wichtigsten Prozeduren des Programms SEC-MPEG in Ablaufdiagrammen skizziert werden.

**Prozedure EncryptMpeg ()**



**Prozedure DecryptSecMpeg ()**



**Abbildung 5: Ablaufdiagramm**

## **5. Ergebnisse**

## 5.1 Integrität

Als Zertifizierungsmechanismus wurde der Cyclic-Redundanz-Check (CRC) in der 16- und 32-bit-Variante gewählt. Dieser Mechanismus errechnet zwar kein 100%ig einzigartiges Zertifikat, hat jedoch die Vorteile der Verfügbarkeit, der schon allgemeine Verbreitung, seiner Schnelligkeit und einfachen Portierbarkeit (16-bit-Code Varianten).

Allerdings ist der schließlich benutzte Algorithmus im SECMPEG-Stream einfach auswechselbar, andere Algorithmen (wie MD2, MD4, MD5) sind möglich.

Da Schnelligkeit oberstes Gebot war, mußten die ersten Konzepte bezüglich des Integritätsschutzes wieder fallen gelassen werden. Jedes Verfahren, daß auf eine Huffman-Dekodierung zwecks Analyse des Streams hinauslief, vervielfältigte den Rechenaufwand. Somit wurden zuerst beide CRC-Varianten implementiert und mit mehreren der verschiedensten MPEG-I-Stream getestet. Die Ergebnisse waren verblüffend.

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
		∅			<b>3.595</b>

**Tabelle 2: CRC - 16 Bit**

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
		∅			<b>7,701</b>

**Tabelle 3: CRC - 32 Bit**

Die ersten beiden (noch nicht optimierten) CRC-Algorithmen produzierten einen zu vernachlässigenden Rechenaufwand (Overhead). Auf einem Intel 386'er PC unter MS-DOS ca. 3.5 Sekunden/MB; im Vergleich dazu dauert die Dekodierung eines MPEG-I-Stream auf dem gleichen Rechner ca. eine Minute. Die 32-bit-Variante brauchte doppelt soviel Rechenzeit, da sie hier durch 16-bit-Code realisiert wurde. Ein 32-bit-Betriebssystem braucht nur unbedeutend länger zur Berechnung eines 32-bit-CRC's als zur Berechnung eines 16-bit-CRC's.

Hiermit zeichnete sich immer weiter ab, daß ein Integritätsschutz des kompletten Stream möglich war. Die Verfahren wurden von nun an nur noch optimiert und an die vorhandenen Input-/Outputprozeduren des zugrundeliegenden Codes angepasst.

## 5.2. Vertraulichkeit

Bei der Entwicklung der Mechanismen für die Vertraulichkeit wurde in erster Line auf eine hohe Performance Wert gelegt. Die Confidentiality-Level 1 und 2 bringen dies zum Ausdruck, hier wurde ausschließlich die Geschwindigkeit in den Vordergrund gestellt. Für Level 1 kommen wir im Durchschnitt auf 10% der Zeit die für das Abspielen des Videoclips benötigt wird. Für Level 2 ist dieser Wert etwas höher, da hier noch ein Teil der Information der Layer 5 & 6 hinzukommt. Im Gegensatz zu den eben genannten Modi steht bei Level 3 nicht die Geschwindigkeit im Vordergrund, sondern die Verschlüsselung aller relevanten Daten. Für die Verschlüsselung benötigen wir im Durchschnitt, über alle getesteten MPEG-Videoclips, die Hälfte der Zeit des MPEG-Players. Die ermittelten Performancewerte sind allerdings nur als Tendenz zu betrachten, da die Geschwindigkeit der Mechanismen jedes MPEG-Streams von folgenden Faktoren bestimmt werden:

- die Anzahl der Intra-, Predicted- und Interpolated Frames
- Kompressionsrate der intra-kodierten Makroblöcke
- verfolgte Strategie bei der Motion Compensation

Für den Dekodiervorgang kann man folgende Tendenz angeben:

- Level 1      30 %
- Level 2      45 %
- Level 3      65 %                      ( 100 % = Kodiervorgang )

Grundlage hierfür war ein MPEG-Stream mit folgender Zusammensetzung:

I P B B B B P B B B B I mit einer Bildfrequenz von 30 Hz, dies entspricht der Empfehlung der ISO-Spezifikation, nach ~ 0.4 sec einen Referenzframe einzufügen.[ISO92]

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	IPPP	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	
			<b>KB / s</b>	<b>3.21</b>	<b>30.02</b>	<b>27.01</b>	<b>5.89</b>	<b>74.75</b>	<b>73.31</b>
FRISCO.MPG	84	IIII	27.4	2.8	3.1	11.5	1	1	
MJ.MPG	619	IIII	96.6	14.7	20.8	86.2	8.3	8.4	
IICM.MPG	1679	IIII	369.1	56.3	82.7	238.7	23.9	24.4	
			<b>KB / s</b>	<b>4.83</b>	<b>32.27</b>	<b>22.34</b>	<b>7.08</b>	<b>71.74</b>	<b>70.47</b>
			<b>KB / s</b>	<b>3.67</b>	<b>30.80</b>	<b>25.01</b>	<b>6.22</b>	<b>73.63</b>	<b>72.25</b>

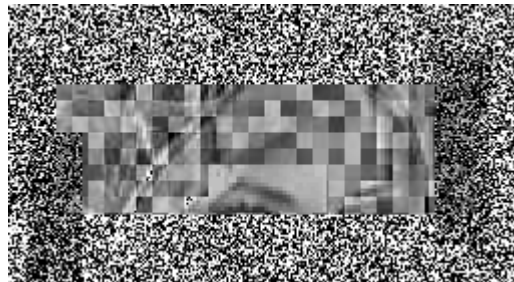
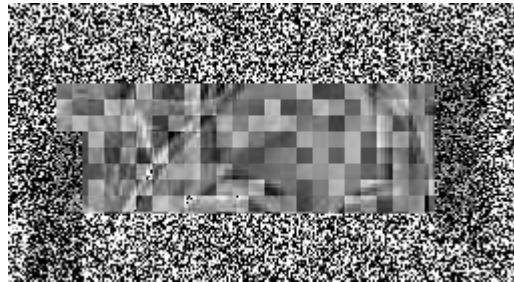
**Tabelle 4: SEC-MPEG Performance**

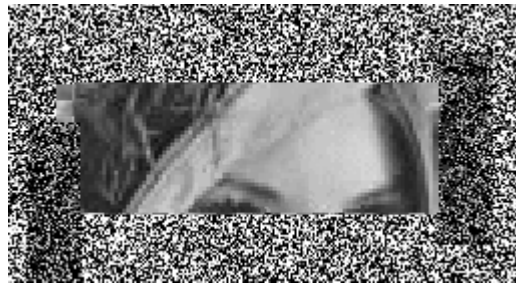
### 5.3. Einschätzung der Sicherheit des Mechanismus

Die Idee auf der Level 1 und 2 gegründet sind, besteht darin, daß wenn die Information der Layer 1-4 verschlüsselt ist, es für einen Angreifer nur mit erheblichen Aufwand möglich ist, einen Teil der Daten auszuwerten. Hier wird zwar jede Verwaltungsinformation verschlüsselt und damit redundante Daten versteckt, aber die Bildinformation ist zu einem hohen Anteil nicht verschlüsselt. Durch sequentielles Suchen eines Startbits für einen Makroblock mit jeweiligem Testdurchlauf pro Bit, kann eine Sequenz von Makroblöcken wiederhergestellt werden. Ein Beispiel ist in Bild 1 - 3 gegeben. Hier wird gute Performance in den Vordergrund gestellt, die Sicherheitsanforderungen werden hier nicht allen Anforderungen gerecht.

In Level 3 werden nun alle intra-kodierten Makroblöcke verschlüsselt. Die Geschwindigkeit des Verschlüsselungsvorganges ist hier von der Zusammensetzung des Streams und der Strategie bei der Kodierung abhängig, (siehe auch 5.3) Da mit den intra-kodierten Blöcken jegliche Referenzen entfallen, kann kein Bildinhalt wieder rekonstruiert werden. Ob mit den z.Z noch nicht geschützten Bewegungsvektoren ein Angreifer eine Deutung auf einen möglichen Bildinhalt vornehmen kann, muß erst noch untersucht werden. Eine Hilfe zur Beurteilung dieser Frage ist mit einem modifizierten MPEG-Player gegeben. [MEY93.3]

Die Einschätzung über die Sicherheit in diesem Modus hängt hier vollständig von der Sicherheit des verwendeten **DES-Modus** ab, hier wird ab einer Blockgröße von 16 Byte im CBC-Modus verschlüsselt, sonst wird der ECB-Modus verwendet.





**Bild 1 - 3 : wiederhergestelltes Teilbild**

möglicher Angriff:

durch sequentielles Suchen werden mehrere Sequenzen von Makroblöcken extrahiert.(Bild 1)  
Da die Master-DC-Koeffizienten nicht wiederherstellbar sind (siehe 3.1), wird versucht für einen Ausschnitt einen sinnvollen Wert für die DC-Koeffizienten zu finden.(Bild 2) Auf Basis dieser Annahme können nun alle weiteren DC-Koeffizienten berechnet werden und damit auch die AC-Koeffizienten.(Bild 3)

## 6. Ausblick

Zielsetzung ist weiterhin die Integration dieser Mechanismen in Sicherheitsdienste, die in der OSI Security Architecture spezifiziert sind, speziell ist an einen Einsatz im Message-Handling-System (MHS / CCITT X.400) gedacht.

Folgende Erweiterungen des Programms SEC-MPEG sind notwendig:

- Erhöhung der Sicherheit der C-Level 1 & 2 (siehe 5.3), bzw eine Zusammenfassung der beiden Level und die Einführung eines neuen Level 2 mit einem höheren Anspruch an die Qualität der Sicherheit. Dies kann mit einer eigenen Entwicklung eines schnellen Stream-Ciphers oder durch Untersuchungen bestehender Alternativen zum DES erfolgen.

Folgende Erweiterungen und Ergänzungen des Programms SEC-MPEG sind noch denkbar:

- Integrität  
Bereitstellung von mehreren zusätzlichen Integritätsalgorithmen (insbesondere MD2, MD4, MD5 )**[RIV91]**. Es sollte auch untersucht werden, inwieweit Verfahren wie z.B RSA einsetzbar sind.
- Vertraulichkeit  
Untersuchungen bestehender Alternativen zum DES :
  - IPES/IDEA           **[MAS90]**
  - FEAL               **[BIH91]**
  - RC2 / RC4       **[FAN92]**und Bereitstellung bei Eignung.
- Ausdehnung der Mechanismen auf die Audio-, System- (Synchronisations-) und User-Data-Spuren, die im MPEG-I-Standard festgeschrieben sind. Hier muß auf die festgelegten CRC-Algorithmen des Audio-Streams Rücksicht genommen werden.
- Für noch hochvolumigere Datenströme müssen ganz neue Mechanismen gefunden werden (MPEG-II bis 10 MB/s !).
- Eine mathematische Untersuchung und Überprüfung des bereitgestellten Mechanismus sollte angefertigt werden .
- Performancesteigerung durch Parallelverarbeitung

Um eine bessere Performance oder eine höhere Qualität der Sicherheit zu erreichen, wäre es sinnvoll diese Mechanismen in MPEG-Encoder und MPEG-Player zu integrieren. Dies erscheint aber nur sinnvoll, wenn die Mechanismen langfristig in die Spezifikation aufgenommen werden.

## Literaturangaben

- [BAD93] Badura, Rolf-Stefan, Meyer-Zajontz, Jörg:  
Quicktime, DVI und MPEG, Referat, Jan. 93
- [BIH91] E. Biham, A. Shamir  
Differential cryptanalysis of Feaf and N-hash  
Eurocrypt '91 Berlin 1991
- [BOR91] Bormann, Ute:  
ISIS 1 Internationale Standards for Informationstechnik - Systemarchitektur,  
Technische Universität Berlin 1991
- [CHI92] Chiariglione, Leonardo: Multimedia Communication (MPEG-II),  
Brussels 1992
- [CT93] c't (Kürzel ku): MPEG-2-Standard festgeklopft, Heft 6, 1993
- [FAN92] Paul Fahn  
FAQ about Today's Cryptographie  
RSA Laboratories 1992
- [GAD93.1] 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
- [GAD93.4] Gadegast Frank:  
Integrität von Multimedia-Daten , Konzeptpapier  
Technische Universität Berlin, Mai / Juni 1993
- [GMD91] Gesellschaft für Mathematik und Datenverarbeitung  
Wolfgang Schneider: SecuDE  
Darmstadt 1991
- [HUN93] Hung, Andy C.:  
PVRG-MPEG-CODEC 1.1, Manual, Stanford, 1993
- [ISO90] 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

- [MAS90] James L. Massey, Xuejia Lai:  
IDEA , International Data Encyrption Standard  
ETH Zurich 1990
  
- [MEY93] Meyer Jürgen:  
Vertraulichkeit von Multimedia-Daten , Konzeptpapier  
Technische Universität Berlin, Mai / Juni 1993
  
- [MFILT] MPEG-Analyser  
Technische Universität Berlin, Juli 1993
  
- [MUS93] Musman, Hans-Georg; Werner, Oliver; Fuchs, Hendrik:  
Kompressionsalgorithmen für interaktive Multimedia-Systeme,  
IT+TI 2/93, Hannover 1993
  
- [NVR93] North Valey Reserach: Digital Media Development Kit, Reference Manual,  
Version 1.0, Jan. 1993
  
- [PAT92] Patel, Ketan; Smith, Brian C.; Rowe, Lawrence A.:  
Performance of a Software MPEG Video Decoder, Berkeley, 1992
  
- [RIV91] Ronald Rivest  
The MD5 Message Digest Algorithm  
MIT 1991