TeleTrusT

Verein zur Förderung der Vertrauenswürdigkeit
von Informations- und Kommunikationstechnik

MailTrusT

Spezifikation

Version 1.1

Stand: 18. Dezember 1996

Fritz Bauspieß
r3 security engineering ag

Folgende TeleTrusT-Mitgliedsfirmen haben durch einen Spendenbeitrag die Ausarbeitung der MailTrusT-Spezfikation ermöglicht:

- Algorithmic Research GmbH
- Bundesamt für Sicherheit in der Informationstechnik
- Competence Center Informatik GmbH
- CompuServe GmbH
- Computer-Communication Network GmbH
- Computer Elektronik Infosys GmbH
- Concord-Eracom Computer Security GmbH
- debis Systemhaus GEI mbH
- Giesecke & Devrient GmbH
- GMD Forschungszentrum für Informationstechnik GmbH
- KryptoKom GmbH
- Nortel Dasa Network Systems GmbH
- ORGA Kartensysteme GmbH
- r^3 security engineering ag
- Siemens AG
- Siemens Nixdorf Informationssysteme AG
- Telenet GmbH

TeleTrusT Deutschland contact points:

TeleTrusT Chairman of the Board:

TeleTrusT Deutschland e.V.
Dr. Albert Glade
c/o Giesecke & Devrient
Postfach 80 07 29
D-81607 München
Telephone +49 (0)89 4119-1856
Telefax +49 (0)89 4119-1396

TeleTrusT Executive Manager:

Prof. Dr. Helmut Reimer
Eichendorffstraße 16
D-99096 Erfurt
Telephone +49 (0)361 34 60 531
Telefax +49 (0)361 34 53 957
e-mail teletrust@t-online.de

Inhaltsverzeichnis

Abkürzungsverzeichnis

ANSI	American National Standards Institute
ASN.1	Abstract Syntax Notation 1
BER	Basic Encoding Rules
CA	Certification Authority
CBC	Cipher Block Chaining (Mode)
CCITT	Comitee Consultatif International Telegraphique et Telecommunication
DEK	Data Encryption Key
DER	Distinguished Encoding Rules
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
ECB	Electronic Codebook (Mode)
EDE	Encryption Decryption Encryption (Mode)
ENC	Encryption
FIPS	Federal Information Processing Standards
FIPS PUB	FIPS Publication
GMT	Greenwich Mean Time
ID	Identification
IETF	Internet Engineering Task Force
IPRA	Internet Policy Registration Authority
ISO	International Organization for Standardization
ITU	International Telecommunication Union
IV	Initialisierungsvektor
KEK	Key Encryption Key
MAC	Message Authentication Code
MD2	Message Digest 2
MD5	Message Digest 5
MIC	Message Integrity Check
MTT	MailTrusT
NA	Naming Authority
PCA	Policy Certification Authority
PEM	Privacy Enhanced Mail
PIN	Personal Identification Number
PKCS	Public-Key Cryptography Standard
PKIX	Public Key Infrastructure (X.509)
PSE	Personal Security Environment
RA	Registration Authority
RFC	Request for Comments
RSA	Rivest, Shamir, Adleman Kryptosystem
SHA	Secure Hash Algorithm
TLCA	Top Level Certification Authority
TTT	TeleTrusT
UTC	Universal Time Code

1 Überblick

Die vorliegende MailTrusT-Spezifikation enthält die folgenden Kapitel:

- Kapitel 1 „Überblick“

  Das vorliegende Kapitel. Gibt einen Überblick über die in dieser Spezifikation enthaltenen Kapitel und ihren Inhalt.

- Kapitel 2 „Einleitung“

  Gibt Hintergrundinformationen zu Entstehung, Zweck und Ziel der MailTrusT-Spezifikation.

- Kapitel 3 „Spezifikation der Dokumentenaustauschformate“

  Enthält die Spezifikation des Austauschformats. Insbesondere werden definiert: die verfügbaren Nachrichtentypen, die durchzuführenden Transformationsschritte sowie Inhalt und Format der in einer Nachricht enthaltenen Kopfzeilen.

- Kapitel 4 „Spezifikation der Zertifizierungsinfrastruktur“

  Definiert Schlüsselmanagement, Sperrlistenmanagement und Zertifikatsformat für MailTrusT-konforme Zertifizierungsinfrastrukturen.

- Kapitel 5 „Festlegung von Namensstrukturen und -formaten“

  Spezifiziert Anforderungen, Vorgaben und Randbedingungen an die Namensgebung innerhalb MailTrusT-konformer Sicherheitsinfrastrukturen.

- Kapitel 6 „Spezifikation der Funktionalität und des Formats der PSE“

  Spezifiziert Inhalt und Funktionalität einer PSE. Nur das äußere Erscheinungsbild einer PSE gegenüber einer Applikation wird definiert. Die Spezifikation der internen Struktur und Funktionalität einer PSE ist nicht Gegenstand der MailTrusT-Spezifikation.

- Kapitel 7 „Festlegung der Kodierung verschiedener Dokumentenformate“

  Nennt die für MailTrusT definierten Dokumentenformate und ihre Identifier.

- Kapitel 8 „Kryptoalgorithmen“

  Spezifiziert die in MailTrusT verwendbaren Kryptoalgorithmen.

- Kapitel 9 „Literaturverzeichnis“

  Referenziert diejenige Literatur, auf Basis derer diese Spezifikation erstellt wurde.

2 Einleitung

Die vorliegende MailTrusT-Spezifikation wurde im Auftrag von TeleTrusT Deutschland erstellt und in Abstimmung mit den Arbeitsgruppen von TeleTrusT Deutschland, insbesondere mit der Projektgruppe MailTrusT erarbeitet.

2.1 Ziel der MailTrusT-Spezifikation

Ziel dieser Spezifikation ist es, eine Basis bereitzustellen, auf der Hersteller Sicherheitskomponenten für sichere E-Mail und sicheren eletronischen Dokumentenaustausch entwickeln können, die zwischen den Herstellern interoperabel sind.

Gerade im Kommunikationsbereich lassen sich proprietäre Sicherheitslösungen kaum noch durchsetzen:

- Es besteht zunehmend Bedarf, auch organisationsübergreifend sicher zu kommunizieren, wobei die Entscheidung über die einzusetzenden Sicherheitskomponenten typischerweise jedoch innerhalb der einzelnen Organisationen gefällt wird.
- Die Verfügbarkeit konkurrierender Produkte wird von Kundenseite verlangt, um die Abhängigkeit von einem einzelnen Hersteller zu vermeiden und um die Kontinuität einer Sicherheitslösung unabhängig von der Entwicklung und zukünftigen Positionierung eines einzelnen Herstellers auch über längere Zeit hinweg sicherstellen zu können.

Die MailTrusT-Spezifikation soll in diesem Zusammenhang als gemeinsame Basis dienen, auf der Sicherheitskomponenten entwickelt werden können, die herstellerübergreifend interoperabel sind.

2.2 Schnittstellen der MailTrusT-Spezifikation

Die MailTrusT-Spezifikation baut überwiegend auf bereits etablierten Standards und Spezifikationen auf, wie PEM ([RFC1421, RFC1422, RFC1423, RFC1424]), X.509 ([X.509]) und PKCS#11 ([PKCS11]). Sie umfaßt unter anderem die Spezifikation folgender Einzelschnittstellen:

- Spezifikation des Dokumentenaustauschformats (Kapitel 3)
- Spezifikation der Zertifizierungsinfrastruktur (Kapitel 4), einschließlich der Spezifikation der Zertifikats- und Sperrlistenformate (Kapitel 4.2.2 und 4.2.3), der Austauschformate für die Kommunikation innerhalb und mit der Zertifizierungsinfrastruktur (Kapitel 4.2.5) und der Regelungen zur Namensvergabe (Kapitel 5).
- Spezifikation der PSE-Schnittstelle (Schnittstelle zu Sicherheitstoken) (Kapitel 6).

2.3 Konformität mit der MailTrusT-Spezifikation

Die MailTrusT-Spezifikation umfaßt mehrere Einzelspezifikationen, die für eine bestimmte Sicherheitskomponente nicht notwendigerweise alle relevant sind. Beispielsweise ist die Spezifikation des Dokumentenaustauschformats für eine Chipkarte als PSE irrelevant, da nicht die Chipkarte, sondern die die Chipkarte verwendende Applikation für die Einhaltung dieses Dokumentenaustauschformats zuständig ist.

Eine Sicherheitskomponente ist dann konform mit dieser Spezifikation, wenn sie alle sie betreffenden Einzelspezifikationen jeweils in vollem Umfang erfüllt. Hierbei gilt jedoch folgende Ausnahme:

- Das in Kapitel 3 „Spezifikation der Dokumentenaustauschformate“ aus Kompatibilität zu PEM ebenfalls spezifizierte symmetrische Schlüsselmanagement (Austausch von Teilnehmerschlüsseln nicht über Zertifikate sondern über symmetrische Master-Schlüssel) ist optional. Wird es realisiert, so ist es so umzusetzen, wie in Kapitel 3 spezifiziert. Eine Realisierung des symmetrischen Schlüsselmanagements ist jedoch für Konformität mit der MailTrusT-Spezifikation nicht erforderlich.

3 Spezifikation der Dokumentenaustauschformate

3.1 Überblick

Das MTT-Dokumentenaustauschformat baut auf PEM auf und unterstützt alle in [RFC1421] spezifizierten Nachrichtentypen. Dies sind die Nachrichtentypen:

- Proc-Type: 4,MIC-ONLY
- Proc-Type: 4,MIC-CLEAR
- Proc-Type: 4,ENCRYPTED

Der ebenfalls in [RFC1421] spezifizierten PEM-Typen „CRL-RETRIEVAL-REQUEST“ und „CRL“ spezifizieren kein Dokumentenaustauschformate, sondern dienen dem Schlüsselmanagement und sind in Kapitel 4 „Spezifikation der Zertifizierungsinfrastruktur“ beschrieben.

Neben den PEM-Nachrichten definiert diese Spezifikation vier weitere, MailTrusT-spezifische Nachrichtentypen:

- Proc-Type: MTT-1,MIC-BIN
- Proc-Type: MTT-1,ENCRYPTED-BIN
- Proc-Type: MTT-1,MIC-RAW
- Proc-Type: MTT-1,ENCRYPTED-RAW

Diese unterscheiden sich von den originären PEM-Nachrichten dadurch, daß keine Kanonisierung der Klartextdaten vorgenommen wird. Mit Hilfe dieser Nachrichtentypen wird es damit im Gegensatz zu PEM möglich, auch binäre Daten direkt zu verarbeiten.

Bei den Typen „MIC-RAW“ und „ENCRYPTED-RAW“ entfällt darüber hinaus die abschließende Kodierung der Nachrichten. Nachrichtenkopf und Nachrichtenrumpf werden getrennt gespeichert, so daß insgesamt diese Nachrichtentypen besonders für die Verarbeitung von Massendaten gut geeignet sind (Keine Expansion der Originaldaten, kein „Umpacken“ der Originaldaten zum Einbau in eine MTT-Nachricht).

In allen vier MTT-Nachrichtentypen sind darüber hinaus zusätzliche Informationen über die in der Nachricht eingebetteten Daten enthalten, die zur Information des Benutzers über den Nachrichteninhalt oder für eine automatische Weiterverarbeitung der Daten genutzt werden können.

3.2 Spezifikation

3.2.1 Nachrichtenklassen

3.2.1.1 PEM-Nachrichten

Die MailTrusT-Spezifikation baut für das Dokumentenaustauschformat auf den PEM-Spezifikationen auf ([RFC1421, RFC1422, RFC1423, RFC1424]).

Alle in [RFC1421] definierten PEM-Nachrichtentypen sind gültige Nachrichten im Sinne dieser Spezifikation. Damit stehen als Austauschformate zur Verfügung:

- Proc-Type: 4,MIC-ONLY für signierte Nachrichten (siehe 3.2.2.1)
- Proc-Type: 4,MIC-CLEAR für signierte Nachrichten mit lesbarem Klartext (siehe 3.2.2.2)
- Proc-Type: 4,ENCRYPTED für signierte und verschlüsselte Nachrichten (siehe 3.2.2.3)

Die PEM-Nachrichtentypen „CRL-RETRIEVAL-REQUEST“ und „CRL“ definieren keine Dokumentenaustauschformate und sind in Kapitel 4 „Spezifikation der Zertifizierungsinfrastruktur“ beschrieben.

Die „Proc-Type“-Angabe besteht aus zwei durch Komma getrennten Komponenten:

- Die erste Komponente ist die Versionsangabe. Die hier angegebene Versionsnummer „4“ verweist auf [RFC1421] und gibt an, daß die vorliegende Nachricht vollständig der dort enthaltenen Spezifikation entspricht.
- Die zweite Komponente nennt den Typ der Nachricht. Aus ihr ist erkennbar, welche Sicherheitsfunktionen auf die Nachricht angewendet wurden.

Der Aufbau der einzelnen Nachrichten ist in den nachfolgenden Abschnitten beschrieben.

Nachrichten ohne digitale Signatur werden von PEM nicht unterstützt.

3.2.1.2 MTT-Nachrichten

Neben den PEM-Nachrichten definiert diese Spezifikation vier weitere MTT-spezifische Nachrichtentypen.

Zur Unterscheidung von den Original-PEM-Nachrichtentypen wird für diese erweiterten Typen die Proc-Type-Version „MTT-1“ spezifiziert (siehe Beschreibung des „Proc-Type“-Feldes im vorangegangenem Kapitel „PEM-Nachrichten“). Die Versionskennung „MTT-1“ verweist auf die vorliegende Version 1 der MailTrusT-Spezifikation.

Durch diese Kennung wird es MailTrusT-kompatiblen Komponenten und Applikationen ermöglicht, zu unterscheiden, ob eine Standard-PEM-Nachricht oder eine erweiterte MTT-Nachricht vorliegt. Reine PEM-Anwendungen werden die Versionskennung „MTT-1“ nicht kennen und haben damit zumindest die Chance, MTT-spezifische Nachrichten als nicht PEM-konform zu erkennen und zurückzuweisen.

Die Versionskennung „MTT-1“ ist außerdem nicht numerisch und wird damit auch in Zukunft nicht mit den Versionskennungen möglicher PEM-Weiterentwicklungen kollidieren, da für PEM eine rein numerische Versionskennung festgelegt ist.

Bei MTT-Nachrichten entfällt grundsätzlich der Schritt der Kanonisierung der Klartextdaten. Binäre Daten können damit im Gegensatz zu PEM mit MTT-Nachrichten direkt verarbeitet werden. Eine Kodierung von binären Daten vor der Bearbeitung ist nicht mehr erforderlich.

MTT-Nachrichten enthalten darüber hinaus in der PEM-Kopfzeile „Content-Domain“ Zusatzinformationen über die in der Nachricht transportierten Daten. Genauere Angaben hierzu sind im Kapitel 3.2.4.2 „Content-Domain“ spezifiziert.

Die vorliegende Spezifikation definiert die folgenden zusätzlichen Nachrichtentypen:

- Proc-Type: MTT-1,MIC-BIN für signierte binäre Nachrichten (siehe 3.2.2.4)
- Proc-Type: MTT-1,ENCRYPTED-BIN für signierte und verschlüsselte binäre Nachrichten (siehe 3.2.2.5)
- Proc-Type: MTT-1,MIC-RAW für signierte Massendaten (siehe 3.2.2.6)
- Proc-Type: MTT-1,ENCRYPTED-RAW für signierte und verschlüsselte Massendaten (siehe 3.2.2.7)

Die Nachrichtentypen „MIC-BIN“ und „ENCRYPTED-BIN“ entsprechen den PEM-Nachrichtentypen „MIC-ONLY“ bzw. „ENCRYPTED“ für binäre Daten.

Die Nachrichtentypen „MIC-RAW“ und „ENCRYPTED-RAW“ entsprechen in etwa dem PEM-Nachrichtentyp „MIC-CLEAR“, da auch hier die abschließende Kodierung der Daten entfällt. Im Gegensatz zu „MIC-CLEAR“ liefern die Transformationen „MIC-RAW“ und „ENCRYPTED-RAW“ in der Regel jedoch echt binäre Daten, die so beispielsweise nicht direkt über Internet-Mail übertragen werden können.

Nachrichten ohne digitale Signatur werden im Rahmen der MailTrusT-Spezifikation auch von den MTT-Nachrichtentypen nicht unterstützt. Es gibt jedoch innerhalb der MailTrusT-Gruppe eine Zusatzvereinbarung zu dieser Spezifikation, in der auch die Verwendung unsignierter Nachrichten geregelt ist. Unsignierte Nachrichten liegen jedoch stets außerhalb der MailTrusT-Spezifikation.

3.2.2 Nachrichtentypen

3.2.2.1 Proc-Type: 4,MIC-ONLY

Mit dem Nachrichtentyp „Proc-Type: 4,MIC-ONLY“ können Textdateien so bearbeitet werden, daß sie authentisch und integer übertragen werden können.

Binäre Daten können mit diesem Nachrichtentyp nicht direkt bearbeitet werden. Hierzu sind die Nachrichtentypen „Proc-Type: MTT-1,MIC-BIN“ (siehe 3.2.2.4) und „Proc-Type: MTT-1, MIC-RAW“ (siehe 3.2.2.6) vorgesehen.

Folgende Transformationsschritte sind zur Erstellung einer „MIC-ONLY“-Nachricht durchzuführen (siehe auch Kapitel 3.2.3 „Transformation von Nachrichten“):

- Kanonisieren (siehe 3.2.3.1)
- Signieren (siehe 3.2.3.2)
- Kodieren (siehe 3.2.3.4)
- Zusammensetzen (siehe 3.2.3.5)

Nachrichten vom Typ „Proc-Type: 4,MIC-ONLY“ sind vollständig kompatibel zu Standard-PEM-Nachrichten des gleichen Typs, sofern nur Kryptoverfahren verwendet wurden, die auch im PEM-Standard [RFC1423] spezifiziert sind.

3.2.2.2 Proc-Type: 4,MIC-CLEAR

Dieser Nachrichtentyp ist nahezu identisch mit den Nachrichtentyp „Proc-Type: 4, MIC-ONLY“.

Der einzige Unterschied zu „MIC-ONLY“-Nachrichten besteht darin, daß der zu übertragende Nachrichteninhalt vor der Versendung nicht kodiert wird und damit im Klartext lesbar bleibt. Die Kodierung der Kopffelder bleibt davon unberührt (siehe Kapitel 3.2.3.4 „Kodieren“).

Transformationsschritte und Eigenschaften sind ansonsten identisch mit denen des Typs „MIC-ONLY“ (siehe 3.2.2.1).

3.2.2.3 Proc-Type: 4,ENCRYPTED

Mit dem Nachrichtentyp „Proc-Type: 4,ENCRYPTED“ können Textdateien so bearbeitet werden, daß sie authentisch, integer und vertraulich übertragen werden können.

Binäre Daten können mit diesem Nachrichtentyp nicht direkt bearbeitet werden. Hierzu sind die Nachrichtentypen „Proc-Type: MTT-1,ENCRYPTED-BIN“ (siehe 3.2.2.5) und „Proc-Type: MTT-1,ENCRYPTED-RAW“ (siehe 3.2.2.7) vorgesehen.

Folgende Transformationsschritte sind zur Erstellung einer „ENCRYPTED“-Nachricht durchzuführen (siehe auch Kapitel 3.2.3 „Transformation von Nachrichten“):

- Kanonisieren (siehe 3.2.3.1)
- Signieren (siehe 3.2.3.2)
- Verschlüsseln (siehe 3.2.3.3)
- Kodieren (siehe 3.2.3.4)
- Zusammensetzen (siehe 3.2.3.5)

Nachrichten vom Typ „Proc-Type: 4,ENCRYPTED“ sind vollständig kompatibel zu Standard-PEM-Nachrichten des gleichen Typs, sofern nur Kryptoverfahren verwendet wurden, die auch im PEM-Standard [RFC1423] spezifiziert sind.

3.2.2.4 Proc-Type: MTT-1,MIC-BIN

Der Nachrichtentyp „Proc-Type: MTT-1,MIC-BIN“ wurde als Erweiterung zu PEM für MailTrusT neu eingeführt. Mit ihm können im Gegensatz zu den Nachrichtentypen „Proc-Type: 4,MIC-ONLY“ (siehe 3.2.2.1) und „Proc-Type: 4,MIC-CLEAR“ (siehe 3.2.2.2) beliebige, insbesondere auch binäre Daten bearbeitet werden.

Der Nachrichtentyp „Proc-Type: MTT-1,MIC-BIN“ sichert wie die Nachrichtentypen „Proc-Type: 4,MIC-ONLY“ und „Proc-Type: 4,MIC-CLEAR“ die Authentizität und Integrität der zu übertragenden Daten.

Nachrichten vom Typ „Proc-Type: MTT-1,BIN“ werden im Gegensatz zu „MIC-CLEAR“-Nachrichten stets kodiert, so daß sie direkt über Mail-Systeme übertragen werden können. Für signierte aber unkodierte Nachrichten steht der Nachrichtentyp „Proc-Type: MTT-1,MIC-RAW“ (siehe 3.2.2.6) zur Verfügung.

Folgende Transformationsschritte sind zur Erstellung einer „MIC-BIN“-Nachricht durchzuführen (siehe auch Kapitel 3.2.3 „Transformation von Nachrichten“):

- Signieren (siehe 3.2.3.2)
- Kodieren (siehe 3.2.3.4)
- Zusammensetzen (siehe 3.2.3.5)

Nachrichten vom Typ „Proc-Type: MTT-1,MIC-BIN“ sind inkompatibel zu Standard-PEM-Nachrichten, da der in PEM generell übliche Transformationsschritt der Kanonisierung bei diesem Nachrichtentyp entfällt.

3.2.2.5 Proc-Type: MTT-1,ENCRYPTED-BIN

Der Nachrichtentyp „Proc-Type: MTT-1,ENCRYPTED-BIN“ wurde ebenfalls als Erweiterung zu PEM für MailTrusT neu eingeführt. Mit ihm können im Gegensatz zum Nachrichtentyp „Proc-Type: 4,ENCRYPTED“ (vgl. 3.2.2.3) beliebige, insbesondere auch binäre Daten bearbeitet werden.

Der Nachrichtentyp „Proc-Type: MTT-1,ENCRYPTED-BIN“ sichert wie der Nachrichtentyp „Proc-Type: 4,ENCRYPTED“ die Authentizität, Integrität und Vertraulichkeit der zu übertragenden Daten.

Folgende Transformationsschritte sind zur Erstellung einer „ENCRYPTED-BIN“-Nachricht durchzuführen (siehe auch Kapitel 3.2.3 „Transformation von Nachrichten“):

- Signieren (siehe 3.2.3.2)
- Verschlüsseln (siehe 3.2.3.3)
- Kodieren (siehe 3.2.3.4)
- Zusammensetzen (siehe 3.2.3.5)

Nachrichten vom Typ „Proc-Type: MTT-1,ENCRYPTED-BIN“ sind inkompatibel zu Standard-PEM-Nachrichten, da der in PEM generell übliche Transformationsschritt der Kanonisierung bei diesem Nachrichtentyp entfällt.

3.2.2.6 Proc-Type: MTT-1,MIC-RAW

Der Nachrichtentyp „Proc-Type: MTT-1,MIC-RAW“ wurde eingeführt, um auch die Bearbeitung von Massendaten im Rahmen dieser Spezifikation möglich zu machen. Er unterscheidet sich von Standard-PEM-Nachrichten in folgenden Punkten:

- ***Keine Kanonisierung der Eingangsdaten***,
  damit können binäre Daten direkt verarbeitet werden.
- ***Keine Kodierung der Ausgangsdaten***,
  damit erfolgt durch die anzuwendenden Transformationen (siehe 3.2.3) keine Expansion der Ausgangsdaten. Die transformierten Daten haben bis auf den zusätzlichen Nachrichtenkopf die gleiche Größe wie die Originaldaten.
- ***Getrennte Speicherung von Nachrichtenkopf und Nachrichtenkörper***,
  damit kann der Nachrichtenkopf im Anschluß an die Bearbeitung der Nachricht separat erzeugt und insbesondere die Signatur im Anschluß an die Bearbeitung des Nachrichteninhalts berechnet und in den Nachrichtenkopf eingetragen werden.

Auf diese Weise wird erreicht, daß Massendaten

- sequentiell verarbeitet werden können (Eintragung der Signatur im Nachrichtenkopf nach Bearbeitung des Nachrichteninhalts) und
- bei der Verarbeitung nur unwesentlich um einen von der Nachrichtenlänge unabhängigen Anteil erweitert werden (Keine Kanonisierung und keine Kodierung, Erweiterung der Daten durch Nachrichtenkopf und Padding typischerweise unter 4 KByte).

Der Nachrichtentyp „Proc-Type: MTT-1,MIC-RAW“ sichert wie die Nachrichtentypen „Proc-Type: 4,MIC-ONLY“, „Proc-Type: 4,MIC-CLEAR“ und „Proc-Type: MTT-1,MIC-BIN“ die Authentizität und Integrität der zu übertragenden Daten.

Folgende Transformationsschritte sind zur Erstellung einer „MIC-RAW“-Nachricht durchzuführen (siehe auch Kapitel 3.2.3 „Transformation von Nachrichten“):

- Signieren (siehe 3.2.3.2)
- Zusammensetzen (siehe 3.2.3.5)

Aufgrund der für diesen Nachrichtentyp nicht durchgeführten Kodierung liegt der Nachrichteninhalt nach der Bearbeitung in der Regel in Form echt binärer Daten vor, zu deren Übertragung ein entsprechendes 8-Bit-transparentes Übertragungsmedium erforderlich ist. Eine direkte Übertragung dieser Nachrichten beispielsweise über Internet-Mail ist in der Regel nicht möglich.

Beim Nachrichtentyp „Proc-Type MTT-1,MIC-RAW“ werden Nachrichtenkopf und Nachrichtenkörper getrennt voneinander gespeichert. Es ist dafür Sorge zu tragen, daß beide Nachrichtenteile an den Empfänger übertragen werden, damit dieser die Nachricht entsprechend auswerten kann.

Nachrichten vom Typ „Proc-Type: MTT-1,MIC-RAW“ sind inkompatibel zu Standard-PEM-Nachrichten, da der in PEM generell übliche Transformationsschritt der Kanonisierung bei diesem Nachrichtentyp entfällt und Nachrichtenkopf und Nachrichtenkörper getrennt gespeichert werden.

3.2.2.7 Proc-Type: MTT-1,ENCRYPTED-RAW

Der Nachrichtentyp „Proc-Type: MTT-1,ENCRYPTED-RAW“ dient wie der Nachrichtentyp „MTT-1,MIC-RAW“ insbesondere für die Verarbeitung von Massendaten und unterscheidet sich von anderen Nachrichtentypen ebenfalls durch die Punkte:

- Keine Kanonisierung der Eingangsdaten
- Keine Kodierung der Ausgangsdaten
- Getrennte Speicherung von Nachrichtenkopf und Nachrichtenkörper

(vgl. 3.2.2.6 „Proc-Type: MTT-1,MIC-RAW“).

Der Nachrichtentyp „Proc-Type: MTT-1,ENCRYPTED-RAW“ sichert wie die Nachrichtentypen „Proc-Type: 4,ENCRYPTED“ und „Proc-Type: MTT-1,ENCRYPTED-BIN“ die Authentizität, Integrität und Vertraulichkeit der zu übertragenden Daten.

Folgende Transformationsschritte sind zur Erstellung einer „ENCRYPTED-RAW“-Nachricht durchzuführen (siehe auch Kapitel 3.2.3 „Transformation von Nachrichten“):

- Signieren (siehe 3.2.3.2)
- Verschlüsseln (siehe 3.2.3.3)
- Zusammensetzen (siehe 3.2.3.5)

Aufgrund der für diesen Nachrichtentyp nicht durchgeführten Kodierung liegt der Nachrichteninhalt nach der Bearbeitung in der Regel in Form echt binärer Daten vor, zu deren Übertragung ein entsprechendes 8-Bit-transparentes Übertragungsmedium erforderlich ist. Eine direkte Übertragung dieser Nachrichten beispielsweise über Internet-Mail ist in der Regel nicht möglich.

Beim Nachrichtentyp „Proc-Type MTT-1,ENCRYPTED-RAW“ werden Nachrichtenkopf und Nachrichtenkörper getrennt voneinander gespeichert. Es ist dafür Sorge zu tragen, daß beide Nachrichtenteile an den Empfänger übertragen werden, damit dieser die Nachricht entsprechend auswerten kann.

Nachrichten vom Typ „Proc-Type: MTT-1,ENCRYPTED-RAW“ sind inkompatibel zu Standard-PEM-Nachrichten, da der in PEM generell übliche Transformationsschritt der Kanonisierung bei diesem Nachrichtentyp entfällt und Nachrichteninhalt und Nachrichtenkopf getrennt gespeichert werden.

3.2.3 Transformation von Nachrichten

Die Transformation von Nachrichten folgt der PEM-Spezifikation nach RFC1421. Folgende Transformationsschritte sind definiert:

1. Kanonisieren

   Die in lokaler Darstellung und Kodierung vorliegende Nachricht wird in eine einheitliche Darstellung umgewandelt.

2. Signieren

   Über die gegebenenfalls nach Schritt 1 kanonisierte Originalnachricht wird eine digitale Signatur berechnet.

3. Verschlüsseln

   Die gegebenenfalls nach Schritt 1 kanonisierte Originalnachricht wird verschlüsselt.

4. Kodieren

   Die nach Schritt 1 bis Schritt 3 bearbeitete Nachricht wird so kodiert, daß sie über Mail-Systeme unbeschadet übertragen werden kann.

5. Zusammensetzen

   Der nach Schritt 4 vorliegende Nachrichtenkörper wird um Kopfzeilen erweitert und durch einen Rahmen von nicht zur Nachricht gehörenden Teilen getrennt, beispielsweise von im Laufe der Übertragung entstehenden Mail-Kopfzeilen oder automatisch angehängten „Signature-Files“.

Diese Transformationsschritte werden je nach Nachrichtentyp nicht immer alle ausgeführt. Die nachfolgende Tabelle gibt Aufschluß darüber, welche Transformationsschritte bei welchen Nachrichtentypen genutzt werden:

	Nachrichtentypen						
Transformationsschritte	MIC-ONLY	MIC-CLEAR	ENCRYPTED	MIC-BIN	ENCRYPTED-BIN	MIC-RAW	ENCRYPTED-RAW
1. Kanonisieren	X	X	X				
2. Signieren	X	X	X	X	X	X	X
3. Verschlüsseln			X		X		X
4. Kodieren	X	(X)[1]	X	X	X	(X)[2]	(X)[2]
5. Zusammensetzen	X	X	X	X	X	X[2]	X[3]

Tabelle 1: Zusammenhang zwischen Nachrichtentypen und Transformationsschritten

---

1 Bei Nachrichten der Typen „MIC-CLEAR“, „MIC-RAW“ und „ENCRYPTED-RAW“ wird keine Kodierung der zu transportierenden Daten vorgenommen. Die in den Kopffeldern enthaltenen binären Informationen, beispielsweise über kryptographische Schlüssel, werden jedoch auch hier für die Nachrichtenübertragung nach dem in 3.2.3.4 beschriebenen Verfahren kodiert.

2 Bei den Nachrichtentypen „MIC-RAW“ und „ENCRYPTED-RAW“ werden die transformierten Nutzdaten im Gegensatz zu allen anderen Nachrichtentypen nicht in der MTT-Nachricht selbst sondern in einer separaten Datei gespeichert. Die Erzeugung und Behandlung der Kopfzeilen bleibt gegenüber den anderen MTT-Nachrichten unverändert.

Die einzelnen Transformationsschritte sind in den nachfolgenden Abschnitten näher beschrieben.

3.2.3.1 Kanonisieren

Das Kanonisieren von Nachrichten dient dazu, Nachrichten vor ihrer Verarbeitung in eine einheitliche Darstellungsform umzuwandeln, so daß die kryptographischen Operationen unabhängig von der auf dem jeweiligen System gültigen lokalen Darstellungsform durchgeführt werden können.

Die Kanonisierung von Nachrichten ist damit eine systemspezifische Transformation. Sie transformiert eine Nachricht beim Senden aus der systemspezifischen in die einheitliche Darstellungsform und beim Empfangen aus der einheitlichen in die systemspezifische Darstellungsform.

Als einheitliche Darstellungsform, auf der die kryptographischen Algorithmen operieren ist festgelegt, daß

- Nachrichten nach US-ASCII codiert werden (s. Tabelle 2) und
- Zeilenenden einheitlich durch die Zeichenkombination <CR><LF> dargestellt werden.

Da durch die Kanonisierung alle Zeilenendekennungen vereinheitlicht und alle Datenbytes auf 7 Bit US-ASCII beschränkt werden, werden binäre Daten durch diese Operation in der Regel zerstört. Binäre Daten können nur in solchen Nachrichtenformaten direkt übertragen werden, in denen der Schritt der Kanonisierung ausgelassen wird, das heißt in den Nachrichtentypen „Proc-Type: MTT-1,MIC-BIN“, „Proc-Type: MTT-1,ENCRYPTED-BIN“, „Proc-Type: MTT-1,MIC-RAW“ und „Proc-Type: MTT-1,ENCRYPTED-RAW“.

Die Kanonisierung der Originaldaten wird durchgeführt bei den Nachrichtentypen

- Proc-Type: 4,MIC-ONLY
- Proc-Type: 4,MIC-CLEAR
- Proc-Type: 4,ENCRYPTED

Zeichensatz der einheitlichen Darstellungsform:

Dez	Oct	Hex	Code	Dez	Oct	Hex	Code	Dez	Oct	Hex	Code	Dez	Oct	Hex	Code
0	000	00	<NUL>	32	040	20	Blank	64	100	40	@	96	140	60	`
1	001	01	<SOH>	33	041	21	!	65	101	41	A	97	141	61	a
2	002	02	<STX>	34	042	22	„	66	102	42	B	98	142	62	b
3	003	03	<ETX>	35	043	23	#	67	103	43	C	99	143	63	c
4	004	04	<EOT>	36	044	24	$	68	104	44	D	100	144	64	d
5	005	05	<ENQ>	37	045	25	%	69	105	45	E	101	145	65	e
6	006	06	<ACK>	38	046	26	&	70	106	46	F	102	146	66	f
7	007	07	<BEL>	39	047	27	'	71	107	47	G	103	147	67	g
8	010	08	<BS>	40	050	28	(	72	110	48	H	104	150	68	h
9	011	09	<HT>	41	051	29	)	73	111	49	I	105	151	69	i
10	012	0A	<LF>	42	052	2A	*	74	112	4A	J	106	152	6A	j
11	013	0B	<VT>	43	053	2B	+	75	113	4B	K	107	153	6B	k
12	014	0C	<FF>	44	054	2C	,	76	114	4C	L	108	154	6C	l
13	015	0D	<CR>	45	055	2D	-	77	115	4D	M	109	155	6D	m
14	016	0E	<SO>	46	056	2E	.	78	116	4E	N	110	156	6E	n
15	017	0F	<SI>	47	057	2F	/	79	117	4F	O	111	157	6F	o
16	020	10	<DLE>	48	060	30	0	80	120	50	P	112	160	70	p
17	021	11	<DC1>	49	061	31	1	81	121	51	Q	113	161	71	q
18	022	12	<DC2>	50	062	32	2	82	122	52	R	114	162	72	r
19	023	13	<DC3>	51	063	33	3	83	123	53	S	115	163	73	s
20	024	14	<DC4>	52	064	34	4	84	124	54	T	116	164	74	t
21	025	15	<NAK>	53	065	35	5	85	125	55	U	117	165	75	u
22	026	16	<SYN>	54	066	36	6	86	126	56	V	118	166	76	v
23	027	17	<ETB>	55	067	37	7	87	127	57	W	119	167	77	w
24	030	18	<CAN>	56	070	38	8	88	130	58	X	120	170	78	x
25	031	19	<EM>	57	071	39	9	89	131	59	Y	121	171	79	y
26	032	1A	<SUB>	58	072	3A	:	90	132	5A	Z	122	172	7A	z
27	033	1B	<ESC>	59	073	3B	;	91	133	5B	[	123	173	7B	{
28	034	1C	<FS>	60	074	3C	<	92	134	5C	\	124	174	7C	|
29	035	1D	<GS>	61	075	3D	=	93	135	5D	]	125	175	7D	}
30	036	1E	<RS>	62	076	3E	>	94	136	5E	^	126	176	7E	~
31	037	1F	<US>	63	077	3F	?	95	137	5F	_	127	177	7F	<DEL>

Tabelle 2: US-ASCII Tabelle

Code	Langform	Bedeutung
ACK	Acknowledge	Bestätigung
BEL	Bell	Glocke, Signal
Blank	Blank	Leerzeichen
BS	Backspace	Schritt zurück, letztes Zeichen vor Cursor löschen
CAN	Cancel	Abbrechen
CR	Carriage Return	Wagenrücklauf, Cursor an den Anfang der Zeile
DC1	Device Control 1	XON, bereit zum Empfangen
DC2	Device Control 2	
DC3	Device Control 3	XOFF, Gegenstück zu XON (DC1): nicht bereit
DC4	Device Control 4	
DEL	Delete	Löschen, erstes Zeichen nach Cursor löschen
DLE	Data Link Escape	
EM	End of Medium	Ende des Mediums
ENQ	Enquiry	Anfrage
EOT	End of Transmission	Ende der Übertragung
ESC	Escape	Ausstieg, Abbruch
ETB	End Transmission Block	
ETX	End of Text	Ende des Texts
FF	Form Feed	Seitenvorschub, neue Seite
FS	File Separator	Datei-Trennzeichen
GS	Group Separator	Gruppen-Trennzeichen
HT	Horizontal Tab	Horizontaler Tabulator
LF	Line Feed	Zeilenvorschub, Cursor eine Zeile tiefer
NAK	Not Acknowledged	Negative Bestätigung
NUL	Null	Das Zeichen ohne Bedeutung
RS	Record Separator	Datensatz-Trennzeichen
SI	Shift In	Umschaltung ein
SO	Shift Out	Umschaltung aus
SOH	Start of Header	Beginn des Kopfes
STX	Start of Text	Beginn des Texts
SUB	Substitute	
SYN	Synchronous Idle	
US	Unit Separator	Einheiten-Trennzeichen

Tabelle 3: Sonderzeichen in der US-ASCII-Tabelle

3.2.3.2 Signieren

Die digitale Signatur, die in diesem Transformationsschritt über die Nachricht berechnet wird, dient der Sicherung von Integrität und Authentizität der Nachricht. Der Empfänger einer MTT-gesicherten Nachricht kann anhand der Signatur prüfen, von wem die Nachricht unterzeichnet und ob sie unverändert übertragen wurde.

Die digitale Signatur wird über die nach 3.2.3.1 kanonisierten Daten berechnet, sofern für den vorgegebenen Nachrichtentyp eine Kanonisierung vorgesehen ist. Andernfalls wird die digitale Signatur direkt über die Original-Daten berechnet. Ein Padding vor Übergabe der Daten an das Signaturverfahren findet nicht statt.

Die Signatur wird bei Verwendung eines asymmetrischen Schlüsselmanagements im Feld „MIC-Info“ (siehe 3.2.4.7), bei Verwendung eines symmetrischen Schlüsselmanagements im Feld „Key-Info“ (siehe 3.2.4.11) im Kopf der Nachricht eingetragen[3].

Im symmetrischen Fall wird für jeden Empfänger der Nachricht ein eigenes „Key-Info“-Feld im Kopf der Nachricht eingetragen, das unter anderem die für ihn spezifisch verschlüsselte digitale Signatur enthält. Bei Verwendung eines asymmetrischen Schlüsselmanagements enthält der Kopf nur ein „MIC-Info“-Feld mit der für den Absender der Nachricht spezifischen digitalen Signatur. Eine genaue Beschreibung der Kopffelder ist in Kapitel 3.2.4 „Kopf-Informationen“ zu finden.

Bei symmetrischem Schlüsselmanagement wird die digitale Signatur stets mit dem gemeinsamen symmetrischen Schlüssel von Sender und Empfänger verschlüsselt.

Im asymmetrischen Fall wird die digitale Signatur nur dann verschlüsselt, wenn auch die Nachricht verschlüsselt wird, d.h. bei den Nachrichtentypen „ENCRYPTED“, „ENCRYPTED-BIN“ und „ENCRYPTED-RAW“. Bei allen anderen Nachrichtentypen wird die digitale Signatur unverschlüsselt in den Kopf der Nachricht eingetragen. (siehe auch Kapitel 3.2.3.3 „Verschlüsseln“).

Die in MailTrusT verwendbaren Signaturalgorithmen sind in Kapitel 8 „Kryptoalgorithmen“ spezifiziert.

Der Transformationsschritt „Signieren“ wird für alle Nachrichtentypen durchgeführt. Nicht signierte Nachrichten werden weder von PEM noch von MailTrusT unterstützt.

3.2.3.3 Verschlüsseln

Die Verschlüsselung der Nachricht dient dem Schutz der Vertraulichkeit der Original-Daten. Durch die Verschlüsselung wird sichergestellt, daß nur der oder die dedizierten Empfänger der Nachricht diese wieder entschlüsseln und auswerten können („Adressierte Vertraulichkeit“).

Die Verschlüsselung wird wie die digitale Signatur über die nach 3.2.3.1 kanonisierte Nachrichtenform berechnet. Wurde aufgrund des Nachrichtentyps keine Kanonisierung durchgeführt, so wird die Verschlüsselung direkt über die Original-Nachricht berechnet. Ein Padding vor Übergabe der Daten an das Verschlüsselungsverfahren findet nicht statt.

Das Ergebnis der Verschlüsselung ersetzt die Original-Nachricht in der zu übertragenden PEM bzw. MTT-Nachricht.

Für jede zu verschlüsselnde Nachricht wird ein neuer zufällig gewählter Nachrichtenschlüssel generiert. Ein Nachrichtenschlüssel darf nur für eine einzige Nachricht verwendet

---

[3] In beiden Fällen handelt es sich um eine echte digitale Signatur auf Basis eines asymmetrischen Kryptoverfahrens. Lediglich das Schlüsselmanagement kann alternativ symmetrisch oder asymmetrisch gelöst werden.

werden. Die Wiederverwendung eines bereits genutzten Nachrichtenschlüssels für eine weitere Nachricht ist unzulässig.

Der Nachrichtenschlüssel wird je Empfänger mit dessen symmetrischen (bei symmetrischem Schlüsselmanagement) bzw. dessen öffentlichen Schlüssel (bei asymmetrischem Schlüsselmanagement) verschlüsselt und in einem entsprechenden „Key-Info“-Feld (siehe 3.2.4.11) im Kopf der Nachricht eingetragen.

Ein Mischen von Empfängern mit symmetrischem und asymmetrischem Schlüsselmanagement ist möglich und zulässig.

Je Empfänger wird ein eigenes „Key-Info“-Feld im Nachrichtenkopf angelegt, das den mit dem Schlüssel des Empfängers verschlüsselten Nachrichtenschlüssel enthält. Es gibt jedoch nur einen Nachrichtenschlüssel. Die Nachricht wird, auch bei einer Adressierung an mehrere Empfänger, nur einmal verschlüsselt und verschickt.

Eine Verschlüsselung von Nachrichten findet statt für die Nachrichtentypen

- Proc-Type: 4,ENCRYPTED
- Proc-Type: MTT-1,ENCRYPTED-BIN
- Proc-Type: MTT-1,ENCRYPTED-RAW

Alle anderen Nachrichtentypen bleiben unverschlüsselt.

Wird die Nachricht verschlüsselt, so wird auch die nach 3.2.3.2 generierte digitale Signatur im Kopf der Nachricht verschlüsselt. Eine Signaturprüfung ist dann nur für die berechtigten Empfänger der Nachricht möglich. Damit soll insbesondere ein Abgleich einer übertragenen verschlüsselten Nachricht gegen eine Liste möglicher Nachrichten durch unberechtigte Dritte verhindert werden.

Die Absenderkennung hingegen bleibt jedoch auch bei verschlüsselten Nachrichten unverschlüsselt im Kopf der MTT-Nachricht allgemein lesbar. Sie ist im Feld „Originator-Certificate“, „Originator-ID-Asymmetric“ oder „Originator-ID-Symmetric“ eingetragen.

Eine genaue Beschreibung der Kopffelder findet sich in Kapitel 3.2.4 „Kopf-Informationen“.

3.2.3.4 Kodieren

PEM und auch MailTrusT wurden vor allem dazu geschaffen, einen sicheren Nachrichtenaustausch über E-Mail zu ermöglichen. Für eine Übertragung über E-Mail müssen signierte bzw. verschlüsselte Nachrichten jedoch bestimmten Beschränkungen in Bezug auf Zeichensatz und maximale Zeilenlänge genügen, da nicht alle Mailsysteme in der Lage sind, binäre Daten direkt zu übertragen. Die in diesem Abschnitt spezifizierte Kodierung dient dazu, beliebige Daten so zu kodieren, daß sie den genannten Beschränkungen genügen.

Die zu kodierenden Daten werden als ein linearer Strom von Datenbytes betrachtet. Die Kodierung von Daten, die sich nicht in ganze Bytes aufteilen lassen, weil am Ende weniger als 8 Bit übrig bleiben, kommt in dem hier spezifizierten Datenaustauschformat nicht vor und wird nicht betrachtet.

Der Datenbytestrom wird sequentiell bearbeitet. Jeweils 3 Byte werden zu einer Bitkette von 24 Bit zusammengefasst, die anschließend wieder in 4 Gruppen zu je 6 Bit aufgeteilt wird. Jede dieser 6-Bit-Gruppen wird gemäß der nachfolgenden Tabelle in ein Codezeichen umgewandelt:

Dez	Oct	Hex	Code	Dez	0ct	Hex	Code	Dez	Oct	Hex	Code	Dez	Oct	Hex	Code
0	000	00	A	16	020	10	Q	32	040	20	g	48	060	30	w
1	001	01	B	17	021	11	R	33	041	21	h	49	061	31	x
2	002	02	C	18	022	12	S	34	042	22	i	50	062	32	y
3	003	03	D	19	023	13	T	35	043	23	j	51	063	33	z
4	004	04	E	20	024	14	U	36	044	24	k	52	064	34	0
5	005	05	F	21	025	15	V	37	045	25	l	53	065	35	1
6	006	06	G	22	026	16	W	38	046	26	m	54	066	36	2
7	007	07	H	23	027	17	X	39	047	27	n	55	067	37	3
8	010	08	I	24	030	18	Y	40	050	28	o	56	070	38	4
9	011	09	J	25	031	19	Z	41	051	29	p	57	071	39	5
10	012	0A	K	26	032	1A	a	42	052	2A	q	58	072	3A	6
11	013	0B	L	27	033	1B	b	43	053	2B	r	59	073	3B	7
12	014	0C	M	28	034	1C	c	44	054	2C	s	60	074	3C	8
13	015	0D	N	29	035	1D	d	45	055	2D	t	61	075	3D	9
14	016	0E	O	30	036	1E	e	46	056	2E	u	62	076	3E	+
15	017	0F	P	31	037	1F	f	47	057	2F	v	63	077	3F	/

Tabelle 4: PEM Kodierungstabelle

Gemäß Voraussetzung sind nur Daten zu kodieren, die sich als Folge vollständiger Bytes darstellen lassen. Es kann jedoch vorkommen, daß am Ende eines Datenstroms weniger als die benötigten 3 Bytes zur Kodierung übrig bleiben. In diesem Fall wird der Datenstrom mit Nullbytes am Ende aufgefüllt, bis auch dort 3 Bytes zur Kodierung zur Verfügung stehen. Darüber hinaus wird als weiteres Codezeichen das ‘=‘ eingeführt, das nicht zur Nachricht gehörende sondern im Rahmen der Codierung angefügte Nullbits kennzeichnet.

Folgende Situationen können am Ende eines Datenstroms auftreten:

- Der Datenstrom läßt sich vollständig in 3-Byte-Gruppen aufteilen:

  Die Byte-Gruppen werden nach obiger Tabelle kodiert. Ein Padding findet nicht statt.

- Es bleiben am Ende zwei Byte übrig:

  Es wird ein Nullbyte angehängt, so daß wieder eine 3-Byte-Gruppe entsteht. Die ersten beiden Codezeichen werden aus den ersten 12 Bit wie üblich erzeugt. Das dritte Codezeichen entsteht aus den letzten 4 Bit des Datenstroms und aus 2 angehängten Nullbits und wird ebenfalls nach obiger Tabelle kodiert. Als viertes Codezeichen wird ein ‘=‘ angehängt, da die zugehörigen Nullbits allesamt durch Padding angefügt wurden.

- Es bleibt am Ende ein Byte übrig:

  Es werden zwei Nullbytes angehängt, so daß wieder eine 3-Byte-Gruppe entsteht. Das erste Codezeichen wird aus den ersten 6 Bit wie üblich erzeugt. Das zweite Codezeichen entsteht aus den letzten 2 Bit des Datenstroms und aus 4 angehängten Nullbits und wird ebenfalls nach obiger Tabelle kodiert. Als drittes und viertes Codezeichen wird jeweils ein ‘=‘ angehängt, da die zugehörigen Nullbits allesamt durch Padding angefügt wurden.

Auf diese Weise läßt sich bei der Dekodierung erkennen, welche Bytes zu den eigentlichen Daten gehören und welche durch Padding bei der Kodierung angehängt wurden:

- Steht kein '=' am Ende der Codefolge, so ließ sich der ursprüngliche Datenstrom vollständig in 3-Byte-Gruppen aufteilen. Es wurden bei der Kodierung keine Nullbytes angehängt.
- Steht ein '=' am Ende der Codefolge, so blieben am Ende des ursprünglichen Datenstroms nach Aufteilung in 3-Byte-Gruppen 2 Byte übrig. Es wurde bei der Kodierung genau 1 Nullbyte angehängt.
- Stehen zwei '=' am Ende der Codefolge, so blieb am Ende des ursprünglichen Datenstroms nach Aufteilung in 3-Byte-Gruppen genau 1 Byte übrig. Es wurden bei der Kodierung zwei Nullbytes angehängt.

Die hier beschriebene Kodierung wird sowohl auf die zu übertragende Nachricht angewendet, als auch auf binäre Daten, die in den Kopffeldern enthalten sind, wie beispielsweise digitale Signaturen oder verschlüsselte Kommunikationsschlüssel. Die Kopffelder sind in Kapitel 3.2.4 „Kopf-Informationen" beschrieben.

Zur Einhaltung der Längenbeschränkung bei der Übertragung von E-Mail wird die entstandene Codefolge wie folgt aufgeteilt:

- Die aus den zu übertragenden Daten entstandenen Codefolge im Körper der Nachricht wird so in Zeilen aufgeteilt, daß jede Zeile bis auf die letzte genau 64 Codezeichen enthält.
- Der Umbruch von Kopfzeilen ist in Kapitel 3.2.4 „Kopf-Informationen" beschrieben.

Die Kodierung wird auf die zu übertragenden Daten angewendet bei den Nachrichtentypen

- Proc-Type: 4,MIC-ONLY
- Proc-Type: 4,ENCRYPTED
- Proc-Type: MTT-1,MIC-BIN
- Proc-Type: MTT-1,ENCRYPTED-BIN

Durch Nichtanwendung der Kodiervorschrift beim Nachrichtentyp „MIC-CLEAR" bleiben dort die zu übertragenden Daten auch nach den Transformationen gemäß dieser Spezifikation im Klartext lesbar.

Durch Nichtanwendung der Kodiervorschrift bei den Nachrichtentypen „MIC-RAW" und „ENCRYPTED-RAW" wird dort eine Expansion der Nutzdaten durch diese Transformation vermieden.

Binäre Daten in den Kopffeldern werden stets, d.h. bei allen PEM- und MTT-Nachrichten, gemäß der hier angegeben Vorschrift kodiert.

3.2.3.5 Zusammensetzen

Nachdem die zu übertragenden Daten nach den oben spezifizierten Regeln transformiert wurden, müssen sie vor einer Übertragung noch zu einer Nachricht zusammengesetzt werden. Dazu werden Begrenzungszeilen hinzugefügt, die die Nachricht von später hinzukommenden Daten, wie beispielsweise Informationen des Mailsystems oder automatisch hizugefügten „Signature-Files" trennen und es werden Kopfzeilen eingefügt, die Zusatzinformationen über die durchgeführten Transformationen, über Schlüssel und über Sender und Empfänger der Nachricht enthalten.

Zur Abgrenzung von Nachrichten werden zwei spezielle Abgrenzungszeilen definiert:

Die Zeile

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
```

bezeichnet den Anfang einer Nachricht und die Zeile

```
-----END PRIVACY-ENHANCED MESSAGE-----
```

bezeichnet das Ende einer Nachricht.

Zwischen der eröffenden Begrenzungszeile und den Nutzdaten werden die benötigten Kopfzeilen eingefügt. Auf diese Weise werden auch die Kopfzeilen gegenüber dem Mailsystem abgegrenzt. Sie werden auf jeden Fall im Mail-Körper und nicht im Mail-Kopf transportiert und auf diese Weise vom Mail-Transportsystem weder interpretiert noch modifiziert.

Kopfzeilen und Nutzdaten werden durch eine Leerzeile voneinander getrennt, d.h. durch die Zeichenfolge <CR><LF><CR><LF>. Leerzeilen zwischen der eröffenden Begrenzungszeile und der ersten Kopfzeilen und innerhalb der Kopfzeilen sind nicht gestattet.

Eine Nachricht hat damit im einfachsten Fall folgenden Aufbau:

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
<Kopfzeilen>
<Leerzeile>
<Transformierte Nutzdaten>
-----END PRIVACY-ENHANCED MESSAGE-----
```

Sofern die Nachrichten gemäß der in 3.2.3.4 spezifizierten Vorgehensweise kodiert sind, können die Begrenzungszeilen nicht zufällig als Teil der Nutzdaten auftreten, da das Zeichen „-“ bei der Kodierung nicht verwendet wird und daher in den Nutzdaten nicht vorkommt. In diesem Fall können Nutzdaten und Begrenzungszeilen stets eindeutig unterschieden werden.

Anders sieht dies bei den Nachrichtentypen „Proc-Type:4,MIC-CLEAR“, „Proc-Type: MTT-1, MIC-RAW“ und „Proc-Type: MTT-1,ENCRYPTED-RAW“ aus, bei denen der Schritt des Kodierens entfällt.

Bei „MIC-CLEAR“-Nachrichten wird in jeder Zeile der Nutzdaten, die mit einem „-“ beginnt, die Zeichenfolge „- “ (<Bindestrich><Leerzeichen>) zu Beginn der Zeile eingefügt. Auf diese Weise wird sichergestellt, daß Nutzdaten und Begrenzungszeilen eindeutig voneinander unterschieden werden können. Die zusätzliche eingefügte Zeichenfolge „- “ wird nach Abschluß aller kryptographischen Transformationen eingefügt und geht nicht in die Berechnung der Signatur der Nachricht ein. Sie ist beim Auswerten der Nachricht vor Anwendung der Kryptoverfahren wieder zu entfernen.

Bei „MIC-RAW“- und „ENCRYPTED-RAW“-Nachrichten werden die transformierten Nutzdaten in eine separate Datei ausgelagert. Es entstehen bei diesen Nachrichtentypen daher zwei Dateien, eine Kopfdatei mit den Kopfzeilen und eine Nachrichtendatei mit den transformierten Nutzdaten. Begrenzungszeilen werden nur in der Kopfdatei verwendet und können damit nicht mit entsprechenden Zeichenketten in der Nachrichtendatei kollidieren. Eine Bearbeitung der Nachrichtendatei durch Einfügen von „- “-Kombinationen ist daher nicht erforderlich und wird bei diesen beiden Nachrichtentypen nicht durchgeführt.

Die Kopfdatei hat bei „MIC-RAW“- und „ENCRYPTED-RAW“-Nachrichten folgenden Aufbau:

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
<Kopfzeilen>
<Leerzeile>
-----END PRIVACY-ENHANCED MESSAGE-----
```

Das Kopffeld „Content-Domain“ (siehe 3.2.4.2) enthält dabei als Querverweis auf die zugehörige Nachrichtendatei neben anderen Informationen den Dateinamen dieser Nachrichtendatei. Die MailTrusT-Software des Empfängers ist damit in der Lage, die Kopfdatei zu bearbeiten und aus der Kopfdatei automatisch zu entnehmen, auf welche Nachrichtendatei sich dieser Kopf bezieht.

Nachrichtendateien von „MIC-RAW“- und „ENCRYPTED-RAW“-Dateien werden in diesem Transformationsschritt nicht weiter bearbeitet. Insbesondere werden keine weiteren Zeichen oder Kopf-, Leer- oder Begrenzungszeilen in diese Nachrichtendateien eingefügt.

In allen Nachrichtentypen können auch mehrere unabhängig voneinander bearbeitete Nutzdatenteile transportiert werden. Jeder Teil erhält seine eigenen Kopfzeilen. Die Teile werden untereinander durch „BEGIN“-Zeilen getrennt. Der letzte Teil wird mit einer „END“-Zeile abgeschlossen.

Eine Folge von drei Nutzdatenteilen kann dann beispielsweise folgendermaßen in einer Nachricht zusammengefasst werden:

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
<Kopfzeilen zu Teil 1>
<Leerzeile>
<Transformierte Nutzdaten zu Teil 1>
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
<Kopfzeilen zu Teil 2>
<Leerzeile>
<Transformierte Nutzdaten zu Teil 2>
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
<Kopfzeilen zu Teil 3>
<Leerzeile>
<Transformierte Nutzdaten zu Teil 3>
-----END PRIVACY-ENHANCED MESSAGE-----
```

Bei „MIC-RAW“- und „ENCRYPTED-RAW“-Nachrichten gilt entsprechendes für die Kopfdateien, jedoch ohne die Nutzdaten, da diese ausgelagert sind (Die <Leerzeile>n bleiben erhalten !) . Die Nachrichtendateien dieser Nachrichtentypen bleiben davon unberührt. In einem vergleichbaren Beispiel für „MIC-RAW“- und „ENCRYPTED-RAW“-Nachrichten würden dann zu einer Kopfdatei drei separate Nachrichtendateien gehören, da die Kopfdatei auf drei separate Nachrichtenteile verweist.

Bei allen Nachrichtentypen ist außerdem eine Schachtelung von Nachrichten in beliebiger Tiefe zulässig, das heißt die Nutzdaten einer Nachricht können selbst wieder eine PEM- oder MTT-Nachricht sein. Es ist im Rahmen dieser Spezifikation jedoch nicht gefordert, daß solche in einer Nachricht eingepackten Nachrichten von MTT-Applikationen automatisch als solche erkannt und wiederum bearbeitet werden.

3.2.4 Kopf-Informationen

Die nachfolgenden Unterkapitel enthalten eine Beschreibung der in PEM- und MTT-Nachrichten verwendbaren Kopfzeilen.

Eine Kopfzeile besteht stets aus dem Namen der Kopfzeile, einem Doppelpunkt und einem oder mehreren durch Kommata getrennten Feldern.

Die Kopfzeilen sollten im Nachrichtenkopf in der hier angegebenen Reihenfolge eingetragen werden, sofern bei den Einzelbeschreibungen nicht anders spezifiziert.

Kopfzeilen von mehr als 64 Zeichen Länge werden in Analogie zu der Spezifikation in RFC822 und unter Berücksichtigung des PEM-spezifischen Formats wie folgt auf mehrere Zeilen umgebrochen:

- Der Doppelpunkt trennt den Namen der Kopfzeile von ihrem Inhalt. Der Inhalt einer Kopfzeile kann aus mehreren Elementen bestehen, die durch Komma getrennt sind.
- Vor und nach dem Doppelpunkt und vor und nach einem Komma können sich kein, ein oder mehrere Leerzeichen befinden.
- Tabulatorzeichen (<HT>) gelten als Leerzeichen.
- Mehrere Leerzeichen können zu einem Leerzeichen zusammengefasst oder auch weggelassen werden.
- Ein Umbruch entsteht durch Einfügen der Zeichenkombination <CR><LF><Blank>, d.h. Fortsetzungszeilen beginnen stets mit einem Leerzeichen.
- Die nutzbare Zeilenlänge wird auf 64 Zeichen je Zeile begrenzt. Das Leerzeichen zu Beginn einer Fortsetzungszeile wird dabei nicht mitgerechnet.
- Ein Umbruch kann überall dort eingefügt werden, wo ein Leerzeichen stehen kann (siehe oben).
- Elemente von bis zu 64 Zeichen Länge dürfen nicht auf mehrere Zeilen umgebrochen werden.
- Elemente von mehr als 64 Zeichen Länge müssen in einer neuen Zeile beginnen und werden nach genau je 64 Zeichen umgebrochen. Damit wird dieses Element auf mehrere Zeilen verteilt, wobei jede der Zeilen bis auf die letzte genau 64 Zeichen des Elements enthält.

Da in der Vorgängerversion des PEM-Standards [RFC1421] alle Kopffelder als Erweiterungsfelder mit einem „X-Präfix versehen waren, wird empfohlen, in ankommenden Nachrichten auch Felder mit „X-“-Präfix zu akzeptieren und das „X-“-Präfix dabei zu ignorieren.

Beispiel: „X-Proc-Type: 4,MIC-CLEAR“ ist äquivalent zu „Proc-Type: 4,MIC-CLEAR“.

Für abgehende Nachrichten dürfen nur Felder ohne „X-“-Präfix erzeugt werden. Die korrekte Verarbeitung von Feldern mit „X-“-Präfix ist keine Bedingung für Konformität mit dieser Spezifikation.

3.2.4.1 Proc-Type

Das Feld „Proc-Type“ spezifiziert den Typ der Nachricht.

Es kommt in jeder Nachricht genau einmal vor und muß als erste Kopfzeile nach der eröffnenden Begrenzungszeile eingetragen werden.

Das Feld „Proc-Type“ enthält 2 Elemente: eine Versionsnummer und eine Typbezeichnung. Die Versionsnummer bezeichnet die Spezifikation, nach der die Nachricht aufgebaut ist. Die Typbezeichnung spezifiziert die durchgeführten Transformationen (vergleiche Kapitel 3.2.3) und damit den genauen Nachrichtentyp.

Die folgenden Versionsnummern sind für MailTrusT zulässig:

4 Verweist auf RFC1421 und bezeichnet PEM-formatkompatible[4] Nachrichten

MTT-1 Verweist auf diese Spezifikation und bezeichnet MTT-spezifische Nachrichten.

Folgende „Proc-Type“-Inhalte sind für MailTrusT zulässig:

4,MIC-ONLY	Bezeichnet zu PEM-„MIC-ONLY“ formatkompatible Nachrichten (siehe 3.2.2.1)
4,MIC-CLEAR	Bezeichnet zu PEM-„MIC-CLEAR“ formatkompatible Nachrichten (siehe 3.2.2.2)
4,ENCRYPTED	Bezeichnet zu PEM-„ENCRYPTED“ formatkompatible Nachrichten (siehe 3.2.2.3)
4,CRL-RETRIEVAL-REQUEST	Bezeichnet PEM-kompatible Nachrichten zum Abruf von Sperrlisten (CRL=Certificate Revocation List) (siehe 4.2.5.3)
4,CRL	Bezeichnet PEM-kompatible Nachrichten zum Transport von Sperrlisten (CRL=Certificate Revocation List) (siehe 4.2.5.4)
MTT-1,MIC-BIN	Bezeichnet MTT-spezifische Nachrichten zur Übertragung beliebiger signierter Daten (siehe 3.2.2.4).
MTT-1,ENCRYPTED-BIN	Bezeichnet MTT-spezifische Nachrichten zur Übertragung beliebiger signierter und verschlüsselter Daten (siehe 3.2.2.5).
MTT-1,MIC-RAW	Bezeichnet MTT-spezifische Nachrichten zur Übertragung beliebiger signierter Daten, insbesondere von Massendaten (siehe 3.2.2.6).
MTT-1,ENCRYPTED-RAW	Bezeichnet MTT-spezifische Nachrichten zur Übertragung beliebiger signierter und verschlüsselter Daten, insbesondere von Massendaten (siehe 3.2.2.7).

3.2.4.2 Content-Domain

Das Feld „Content-Domain“ liefert Informationen über den Inhalt der Nachricht.

Diese Kopfzeile ist in jeder Nachricht als zweite Kopfzeile direkt nach der „Proc-Type“-Angabe einzutragen.

Für PEM-Nachrichten, d.h. für Nachrichten mit „Proc-Type“-Versionsnummer „4“ (siehe 3.2.4.1) ist der Wert dieses Kopffeldes auf den String „RFC822“ festgelegt. Nur RFC822-kompatible Inhalte dürfen mit diesen Nachrichtentypen transportiert werden.

Für MTT-Nachrichten mit „Proc-Type“-Versionsnummer „MTT-1“ erhält dieses Feld eine erweiterte Bedeutung und ist wie folgt aufgebaut:

Content-Domain: <data-type>,<compression>,<source-file-name>,<body-file-name>

---

[4] Nachrichten nach dieser Spezifikation mit Versionsnummer „4“ entsprechen von Aufbau und Format exakt den in RFC1421 spezifizierten Nachrichten. Für MailTrusT sind jedoch über den Umfang von RFC1423 hinaus weitere Kryptoverfahren spezifiziert (siehe Kapitel 8), die auch für Nachrichten mit Versionsnummer „4“ verwendet werden dürfen. Eine Nachricht mit Versionsnummer „4“ ist daher genau dann PEM-kompatibel, wenn nur Kryptoverfahren nach RFC1423 verwendet wurden. Andernfalls ist sie lediglich „formatkompatibel“.

<data-type> bezeichnet den Typ der in der Nachricht enthaltenen Daten (MS-Word, Excel, Text, RTF, ...). Die im Rahmen dieser Spezifikation zulässigen Typen sind in Kapitel 7 „Festlegung der Kodierung verschiedener Dokumentenformate“ definiert.

<compression> gibt an, welches Kompressionsverfahren vor Bearbeitung der Daten auf die Original-Datei angewendet wurde. Die Kompression muß beim Versenden vor der ersten der in Kapitel 3.2.3 „Transformation von Nachrichten“ spezifizierten Transformationen durchgeführt werden. Folgende Werte sind für das Feld <compression> im Rahmen dieser Spezifikation definiert:

gzip	GNU Zip
pkzip	PC PKZip
compress	UNIX compress

Andere Werte in diesem Feld sind zulässig und jeweils geeignet abzustimmen.

Die Kompressionsverfahren selbst sind nicht Teil der MailTrusT-Spezifikation. Die Bereitstellung von Kompressionsverfahren ist zum Erreichen der Konformität zu dieser Spezifikation nicht erforderlich.

<source-file-name> enthält den zu den Daten gehörigen Original-Dateinamen. Eine Beschränkung auf bestimmte Typen von Dateinamen (MS-DOS, UNIX, VMS, ...) findet nicht statt. Dateinamen dürfen jedoch keine Kommata enthalten, um eine Verwechslung mit den Kommata zur Feldtrennung innerhalb der Kopfzeile zu vermeiden.

<body-file-name> enthält bei „MIC-RAW“- und „ENCRYPTED-RAW“-Nachrichten den Namen der zugehörigen Nachrichtendatei (siehe 3.2.3.5 „Zusammensetzen“). Bei allen anderen Nachrichtentypen bleibt das Feld leer. Bei „MIC-RAW“- und „ENCRYPTED-RAW“-Nachrichten ist die Angabe dieses Feldes optional. Es kann entfallen, wenn die Zuordnung zwischen Kopfdatei und Nachrichtendatei auf andere Weise geeignet umgesetzt wird. Die Dateinamenkonventionen für <source-file-name> (siehe oben) gelten für <body-file-name> entsprechend.

Die Felder <data-type>, <compression>, <source-file-name> und <body-file-name> sind jeweils optional. Die trennenden Kommata müssen jedoch in jedem Fall angegeben werden, damit aus der Postion eines Feldes auf seine Bedeutung geschlossen werden kann.

Keines der Felder darf Kommata enthalten, um eine Verwechslung mit den Kommata zur Feldtrennung innerhalb der Kopfzeile zu vermeiden.

Fehlt das Feld <compression>, so bedeutet das, daß die Daten nicht komprimiert wurden.

Beispiele:

- MS-Word-6 Datei, komprimiert mit GNU Zip in einer „MIC-BIN“-Nachricht:

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Proc-Type: MTT-1,MIC-BIN
Content-Domain: MS-Word-6,gzip,fritz.doc,
...
```

- Dieselbe MS-Word-6 Datei, unkomprimiert in einer „MIC-RAW“-Nachricht:

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Proc-Type: MTT-1,MIC-RAW
Content-Domain: MS-Word-6,,fritz.doc,fritz.doc
...
```

- Binäre Datei ohne weitere Angaben in einer „ENCRYPTED-RAW“-Nachricht:

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Proc-Type: MTT-1,ENCRYPTED-RAW
Content-Domain:,,,xray.raw
...
```

- Binäre Datei ohne weitere Angaben in einer „ENCRYPTED-BIN“-Nachricht:

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Proc-Type: MTT-1,ENCRYPTED-BIN
Content-Domain:,,,
...
```

3.2.4.3 DEK-Info

Das „DEK-Info“-Feld gibt an, mit welchem Kryptoverfahren die Nutzdaten bei den Nachrichtentypen „Proc-Type: 4,ENCRYPTED“, „Proc-Type: MTT-1,ENCRYPTED-BIN“ und „Proc-Type: MTT-1,ENCRYPTED-RAW“ verschlüsselt wurden.

Die verfügbaren Verfahren und die im „DEK-Info“-Feld für die einzelnen Verfahren zusätzlich einzutragenden Parameter sind in Kapitel 8.2 „Verschlüsseln von Daten“ spezifiziert.

3.2.4.4 Originator-ID-Asymmetric

Das Feld „Originator-ID-Asymmetric“ gibt bei Verwendung eines asymmetrischen Schlüsselmanagements den Absender der Nachricht an.

Das „Originator-ID-Asymmetric“-Feld ist nur dann vorhanden, wenn kein entsprechendes „Originator-Certificate“-Feld im Nachrichtenkopf eingetragen ist.

Eine Nachricht enthält in der Regel höchstens ein „Originator-ID-Asymmetric“-Feld. Eine Kombination mit weiteren „Originator-ID-Symmetric“-Feldern ist möglich, beispielsweise wenn in einer Nachricht an mehrere Empfänger sowohl Empfänger mit asymmetrischen als auch mit symmetrischem Schlüsselmanagement adressiert werden.

Das „Originator-ID-Asymmetric“-Feld hat folgenden Aufbau:

Originator-ID-Asymmetric: <issuing authority>,<version/expiration>

<issuing authority> gibt diejenige Stelle („issuing authority“) an, die das Zertifikat für den Absender der Nachricht ausgegeben hat, d.h. dessen CA. In dem <issuing authority>-Feld wird der X.500-Distinguished-Name dieser Stelle eingetragen, kodiert nach ASN.1 DER und dem in Kapitel 3.2.3.4 spezifizierten Verfahren.

<version/expiration> enthält die Seriennummer des Zertifikats des Absenders in hexadezimaler Kodierung.

3.2.4.5 Originator-ID-Symmetric

Das „Originator-ID-Symmetric“-Feld bezeichnet den Absender einer Nachricht bei Verwendung eines symmetrischen Schlüsselmanagements.

Dieses Feld hat folgenden Aufbau:

Originator-ID-Symmetric: <entity>,<issuing authority>,<version/expiration>

<entity> enthält den Namen des Absenders. Der Name wird in der Regel als Internet-E-Mail-Adresse in der Form <user>@<domain-qualified-host> angegeben.

<issuing authority> bezeichnet die schlüsselausgebende Instanz. Weitere Festlegungen in Bezug auf dieses Feld werden nicht spezifiziert und sind jeweils geeignet zu vereinbaren.

<version/expiration> dient dazu, bei Verfügbarkeit mehrerer möglicher symmetrischer Schlüssel des Absenders, eindeutig den tatsächlich verwendeten Schlüssel zu kennzeichnen. Die Vergabe einer Seriennummer ist hierfür ausreichend. Zusätzliche Zeitinformationen können in dieses Feld eingebettet werden, um beispielsweise die Gültigkeitsdauer eines Schlüssels zu kennzeichnen. Das Format dieses Feldes wird nicht näher spezifiziert und kann auch innerhalb eines Systems für verschiedene <issuing authorities> unterschiedlich gewählt werden. Eine geeignete Festlegung ist jeweils zu treffen.

Die Felder <issuing authority> und <version/expiration> sind optional und können weggelassen werden, sofern die benötigten Informationen aus den zugehörigen „Recipient-ID-Symmetric“-Feldern entnommen werden können. Die Kommata sind stets anzugeben, damit aus der Position eines Feldes auf seine Bedeutung geschlossen werden kann.

3.2.4.6 Originator-Certificate

Das „Originator-Certificate“-Feld wird nur bei asymmetrischem Schlüsselmanagement verwendet und enthält das Zertifikat des Senders.

Es hat folgenden Aufbau:

Originator-Certificate: <certificate>

<certificate> ist das X.509-Zertifikat des Senders und wird nach ASN.1 DER und dem in 3.2.3.4 beschriebenen Verfahren kodiert.

Die ASN.1-Struktur des Zertifikats ist in Kapitel 4.2.2 „Zertifikate“ beschrieben.

3.2.4.7 MIC-Info

Das „MIC-Info“-Feld wird nur angegeben, wenn für mindestens einen Empfänger ein asymmetrisches Schlüsselmanagement verwendet wird und enthält dann die digitale Signatur der Nachricht.

Das „MIC-Info“-Feld kommt in einer Nachricht höchstens einmal vor[5] und hat folgenden Aufbau:

MIC-Info: <mic-algorithm>,<sig-algorithm>,<digital signature>

<mic-algorithm> bezeichnet den Algorithmus, der zur Berechnung der kryptographischen Prüfsumme (= Hashwert) über die zu signierenden Daten verwendet wurde.

<sig-algorithm> nennt den Algorithmus, der zum Signieren der nach <mic-algorithm> berechneten Prüfsumme verwendet wurde.

<digital signature> schließlich enthält die digitale Signatur über den Nachrichteninhalt, d.h. das Ergebnis aus der Anwendung von <mic-algorithm> und <sig-algorithm>. Die zunächst als binäres Ergebnis entstehende digitale Signatur wird nach dem in 3.2.3.4 spezifizierten Verfahren kodiert.

Die zulässigen Algorithmen und Algorithmen-Identifier für <mic-algorithm> und <sig-algorithm> sind in Kapitel 8 „Kryptoalgorithmen“ definiert.

Bei Nachrichten vom Typ „Proc-Type: 4,ENCRYPTED“, „Proc-Type: MTT-1,ENCRYPTED-BIN“ und „Proc-Type: MTT-1,ENCRYPTED-RAW“ wird die digitale Signatur vor der Kodierung ausserdem noch nach demselben Verfahren verschlüsselt, mit dem auch die Nachricht verschlüsselt wurde. Ziel ist es, auf diese Weise Vergleichsattacken zu verhindern, bei

---

5 Im Gegensatz zu PEM wird im Rahmen dieser Spezifikation nur ein Absender und dementsprechend nur ein MIC-Info-Feld je Nachricht zugelassen.

denen ein Angreifer versucht, übertragene Nachrichten anhand ihrer Signatur gegen eine ihm bekannte Liste möglicher Nachrichten abzugleichen und so zu identifizieren.

Ein „MIC-Info“-Feld darf im Kopf einer Nachricht erst nach dem zugehörigen „Originator-ID-Asymmetric“- bzw. „Originator-Certificate“- und den „Issuer-Certificate“-Feldern angegeben werden. Es gilt für alle nachfolgend angegebenen Empfänger („Recipient-ID-*“-Felder).

3.2.4.8 Issuer-Certificate

Dieses Feld wird nur bei asymmetrischem Schlüsselmanagement verwendet und kann in einem Nachrichtenkopf mehrfach auftreten.

Die „Issuer-Certificate“-Felder dienen dazu, die Überprüfung des im „Originator-Certificate“ übermittelten öffentlichen Absenderschlüssels anhand einer Zertifikatskette zu ermöglichen. (siehe auch Kapitel 4.2.1 „Schlüsselmanagement“)

Jedes „Issuer-Certificate“-Feld enthält ein Zertifikat aus dem zum Absender gehörenden Zertifizierungspfad, d.h. ein Zertifikat, dessen in ihm enthaltener öffentlicher Schlüssel dazu benötigt wird, ein anderes Zertifikat auf Gültigkeit zu überprüfen.

Das „Originator-Certificate“ ist ein Zertifikat über den öffentlichen Schlüssel des Absenders. Das nächste „Issuer-Certificate“ ist ein Zertifikat über den öffentlichen Schlüssel, der zur Überprüfung des „Originator-Certificate“ benötigt wird. Jedes darauf folgende „Issuer-Certificate“ ist ein Zertifikat über den öffentlichen Schlüssel, der zur Überprüfung des jeweils vorausgegangenen „Issuer-Certificate“ benötigt wird. Diese Kette setzt sich fort bis zu einem „Issuer-Certificate“, das direkt mit dem beim Empfänger bekannten Wurzelschlüssel der Zertifizierungshierarchie überprüft werden kann. Auf diese Weise kann der Empfänger einer Nachricht ausgehend vom Wurzelschlüssel über die Kette der „Issuer-Certificate“-Felder und abschließend über das „Originator-Certificate“ direkt die Gültigkeit des im „Originator-Certificate“ enthaltenen öffentlichen Schlüssels des Absenders überprüfen.

Die „Issuer-Certificate“-Felder sollen im Nachrichtenkopf nach dem zugehörigen „Originator-Certificate“-Feld und in aufsteigender Reihenfolge vom „Originator-Certificate“ zur Wurzel der Zertifizierungshierarchie aufgelistet werden. Im Nachrichtenkopf steht dann als nächstes „Issuer-Certificate“ jeweils das Zertifikat zur Verfügung, das zur Überprüfung des direkt vorangehenden Zertifikats benötigt wird. Dies erleichtert die Überprüfung des Zertifizierungspfads. Eine Applikation darf sich jedoch nicht darauf verlassen, daß die „Issuer-Certificate“-Felder genau in dieser Reihenfolge aufgelistet sind, sondern muß auch in der Lage sein, „Issuer-Certificate“-Felder in beliebiger Reihenfolge korrekt auszuwerten.

Wie das „Originator-Certificate“-Feld enthält auch das „Issuer-Certificate“-Feld als einzigen Parameter das in ihm zu übermittelnde und nach ASN.1 DER und dem in 3.2.3.4 beschriebenen Verfahren kodierte Zertifikat. Sein Aufbau lautet:

Issuer-Certificate: <certificate>

Die durch die „Issuer-Certificate“-Felder mitgelieferte Zertifikatskette kann auch anstatt bis zur Wurzel der Hierarchie nur auf eine Zertifikatskette bis zu einer gemeinsamen Teilbaumwurzel von Sender und Empfänger beschränkt werden, die der Empfänger dann anhand seiner eigenen Zertifikatskette verifizieren kann.

3.2.4.9 Recipient-ID-Asymmetric

Das „Recipient-ID-Asymmetric“-Feld wird im Nachrichtenkopf einmal für jeden Empfänger eingetragen, für den ein asymmetrisches Schlüsselmanagement genutzt wird. Es ist logisch unabhängig von dem „Originator-ID-Asymmetric“-Feld, sollte im Nachrichtenkopf jedoch erst nach allen „Originator“-Angaben eingetragen werden.

Das „Recipient-ID-Asymmetric“-Feld hat folgenden Aufbau:

Recipient-ID-Asymmetric: <issuing authority>,<version/expiration>

<issuing authority> gibt diejenige Stelle („issuing authority“) an, die das Zertifikat für den durch diese Kopfzeile gekennzeichneten Empfänger ausgegeben hat, d.h. dessen CA. In dem <issuing authority>-Feld wird der X.500-Distinguished-Name“ dieser Stelle eingetragen, kodiert nach ASN.1 DER und dem in Kapitel 3.2.3.4 spezifizierten Verfahren.

<version/expiration> enthält die Seriennummer des Zertifikats des Empfängers in hexadezimaler Kodierung.

3.2.4.10 Recipient-ID-Symmetric

Das „Recipient-ID-Symmetric“-Feld wird im Nachrichtenkopf einmal für jeden Empfänger eingetragen, für den symmetrisches Schlüsselmanagement genutzt wird.

Es bezieht sich logisch stets auf das letzte vorangegangene „Originator-ID-Symmetric“-Feld und darf im Nachrichtenkopf erst nach diesem eingetragen werden.

Das „Recipient-ID-Symmetric“-Feld hat folgenden Aufbau:

Recipient-ID-Symmetric: <entity>,<issuing authority>,<version/expiration>

<entity> enthält den Namen des durch diese Kopfzeile gekennzeichneten Empfängers. Der Name wird in der Regel als Internet-E-Mail-Adresse in der Form <user>@<domain-qualified-host> angegeben.

<issuing authority> bezeichnet die schlüsselausgebende Instanz. Weitere Festlegungen in Bezug auf dieses Feld werden nicht spezifiziert und sind jeweils geeignet zu vereinbaren.

<version/expiration> dient dazu, bei Verfügbarkeit mehrerer möglicher symmetrischer Schlüssel eindeutig den tatsächlich verwendeten Schlüssel zu kennzeichnen. Die Vergabe einer Seriennummer ist hierfür ausreichend. Zusätzliche Zeitinformationen können in dieses Feld eingebettet werden, um beispielsweise die Gültigkeitsdauer eines Schlüssels zu kennzeichnen. Das Format dieses Feldes wird nicht näher spezifiziert und kann auch innerhalb eines Systems für verschiedene <issuing authorities> unterschiedlich gewählt werden. Eine geeignete Festlegung ist jeweils zu treffen.

3.2.4.11 Key-Info

Das „Key-Info“-Feld enthält Schlüsselinformationen für jeweils einen bestimmten Empfänger der Nachricht und wird je Empfänger im Nachrichtenkopf eingetragen.

Das „Key-Info“- Feld bezieht sich stets auf das letzte vorangegangene „Recipient-ID-*“-Feld und wird im Nachrichtenkopf normalerweise direkt nach diesem eingetragen. Darüber hinaus wird empfohlen, auch für den Absender der Nachricht ein „Key-Info“-Feld bereitzustellen, das nach dem „Originator-ID-*“- bzw. „Originator-Certificate“-Feld und vor den zugehörigen „Recipient-ID-*“-Feldern eingetragen wird. Die Verfügbarkeit eines solchen „Originator“-bezogenen „Key-Info“-Feldes ist jedoch im Rahmen dieser Spezifikation nicht zwingend vorgeschrieben.

Bei **symmetrischem** Schlüsselmanagement hat das „Key-Info“-Feld folgenden Aufbau:

Key-Info: <enc-algorithm>,<mic-algorithm>,<dek>,<mic>

<enc-algorithm> bezeichnet das symmetrische Verschlüsselungsverfahren, mit dem die Felder <dek> und <mic> verschlüsselt wurden. Der Nachrichteninhalt wird mit dem im <dek>-Feld enthaltenen Nachrichtenschlüssel und dem im Kopffeld „DEK-Info“ angegebenen Verschlüsselungsverfahren verschlüsselt. Der Nachrichtenschlüssel im <dek>-Feld und die Prüfsumme im <mic>-Feld werden mit dem durch das vorangegangene „Recipient-ID-Symmetric“-Feld spezifizierten symmetrischen Schlüssel als Key-Encryption-Key (KEK)

verschlüsselt. Die im Rahmen dieser Spezifikation zulässigen Verschlüsselungsverfahren sind in Kapitel 8.3 „Schlüsselmanagement“ spezifiziert.

<mic-algorithm> nennt das zur Berechnung der Prüfsumme (= Hashwert) verwendete Verfahren. Zulässige Verfahren und Identifier sind in Kapitel 8.1 „Hashen von Daten“ spezifiziert. Die Prüfsumme wird nach der gegebenenfalls anzuwendenden Kanonisierung (siehe 3.2.3.1) über die zu übermittelnden Klartextdaten berechnet und mit dem in dem vorangegangenen „Recipient-ID-Symmetric“-Feld referenzierten Schlüssel verschlüsselt im <mic>-Feld abgelegt.

<dek> enthält den mit dem KEK verschlüsselten Nachrichtenschlüssel (s.o. „<enc-algorithm>“) und wird nach dem in 3.2.3.4 spezifizierten Verfahren kodiert.

<mic> enthält die mit dem KEK verschlüsselte Prüfsumme der Nachricht (Hashwert) (s.o. „<mic-algorithm>“) und wird ebenfalls nach dem in 3.2.3.4 spezifizierten Verfahren kodiert.

Bei **asymmetrischem** Schlüsselmanagement hat das „Key-Info“-Feld folgenden Aufbau:

Key-Info: <enc-algorithm>,<dek>

<enc-algorithm> bezeichnet das asymmetrische Verschlüsselungsverfahren, mit dem das Feld <dek> verschlüsselt wurde. Der Nachrichteninhalt wird mit dem im <dek>-Feld enthaltenen Nachrichtenschlüssel und dem im Kopffeld „DEK-Info“ angegebenen Verschlüsselungsverfahren verschlüsselt. Der Nachrichtenschlüssel im <dek>-Feld wird mit dem durch das vorangegangene „Recipient-ID-Asymmetric“-Feld spezifizierten asymmetrischen öffentlichen Schlüssel des Empfängers als Key-Encryption-Key (KEK) verschlüsselt. Die im Rahmen dieser Spezifikation zulässigen Verschlüsselungsverfahren sind in Kapitel 8.3 „Schlüsselmanagement“ spezifiziert.

4 Spezifikation der Zertifizierungsinfrastruktur

4.1 Überblick

Für MailTrusT wird hiermit die Verwendung einer Zertifizierungsinfrastruktur festgelegt, wie sie für PEM in [RFC1422] spezifiziert ist. Eine umfassende Darstellung dieser Infrastruktur ist in der „MailTrusT Infrastrukturbeschreibung“ [Hues95] zu finden.

Die MailTrust-Spezifikation legt in diesem Zusammenhang nur die technische Struktur dieser Zertifizierungsinfrastruktur fest. Es finden im Rahmen dieser Spezifikation keine Zuordnungen zu realen Einrichtungen statt, wie dies in RFC1422 beispielsweise für die Internet Policy Registration Authority (IPRA) der Fall ist. Der Begriff IPRA wird im Rahmen der MailTrusT-Spezifikation durch den Begriff TLCA (Top-Level CA) ersetzt.

Die Zuordnung logischer Infrastrukturelemente zu realen Organisationseinheiten ist für die Anwendungsumgebung zusammen mit einer entsprechenden Security Policy jeweils geeignet festzulegen.

In der vorliegenden Version 1 dieser Spezifikation wird für MailTrusT die Verwendung von X.509-Version-1-Zertifikaten (kurz: X.509v1-Zertifikaten) spezifiziert. Um trotz Verwendung von X.509v1 eine Unterscheidung zwischen Signatur- und Verschlüsselungsschlüsseln zu ermöglichen, ist den Algorithmenidentifiern in Kapitel 8 „Kryptoalgorithmen“ zusätzlich der Verwendungszweck zugeordnet.

Damit ist es auf Basis der MailTrusT-Spezifikation möglich, innerhalb einer Sicherheitsinfrastruktur Signatur- und Verschlüsselungsschlüssel voneinander zu trennen.

Für die vorliegende Version 1 der MailTrusT-Spezifikation gilt dabei jedoch aufgrund der Verwendung PEM-orientierter Nachrichtenformate (s. Kapitel 3) folgende Einschränkung[6]:

In den Nachrichten selbst können Zertifikate von reinen Verschlüsselungsschlüsseln nicht transportiert werden. Das „Originator-Certificate“-Feld enthält stets das Zertifikat für den jeweiligen Signaturschlüssel, der gegebenenfalls auch für Verschlüsselungszwecke verwendbar sein kann. Ein eigenes Feld für reine Verschlüsselungszertifikate[7] ist in PEM nicht vorgesehen und wurde im Rahmen dieser Spezifikation nicht eingeführt, da dies eine grundlegende Änderung des Nachrichtenformats bedeutet hätte.

Eine Trennung von Signatur- und Verschlüsselungsschlüssel ist im Rahmen der Version 1 der MailTrusT-Spezifikation daher nur möglich, wenn für die Verteilung von Verschlüsselungszertifikaten ein eigener Mechanismus bereitgestellt wird.

Für Signatur- und Verschlüsselungsschlüssel können teilweise oder vollständig getrennte Zertifizierungsinfrastukturen aufgebaut werden.

Eine Unterstützung der Trennung von Signatur- und Verschlüsselungsschlüsseln ist für Konformität mit der vorliegenden Spezifikation nicht erforderlich.

---

6 Es wird angestrebt, die MailTrusT-Spezifikation so fortzuschreiben, daß diese Einschränkung entfällt.

7 Verschlüsselungszertifikat: Zertifikat über einen öffentlichen Schlüssel, der für die Verschlüsselung von Nachrichten gedacht ist.

4.2 Spezifikation

4.2.1 Schlüsselmanagement

Für MailTrusT wird ein asymmetrisches Schlüsselmanagement spezifiziert, wie es in [RFC1422] und [Hues95] beschrieben ist.

Es werden jedoch ausschließlich technische Strukturen und Möglichkeiten spezifiziert. Die Abbildung der hier definierten Zertifizierungsinfrastruktur in eine reale Organisation und die Regelung des Umgangs mit Vorgängen und Komponenten sind für ein konkretes System in einer entsprechenden Infrastrukturbeschreibung und einer Security Policy jeweils geeignet festzulegen.

Jeder Teilnehmer einer MailTrusT-kompatiblen Sicherheitsinfrastruktur erhält ein oder mehrere asymmetrische Schlüsselpaare, die er selbst generiert oder von einer CA ausgehändigt bekommt. Die zugehörigen öffentlichen Schlüssel werden mit einer Hierarchie von Zertifikaten so abgesichert, daß sie offen im Netz übertragen und dennoch von jedem anderen Teilnehmer auf Gültigkeit und Unversehrtheit überprüft werden können.

Für MailTrusT wird dazu die folgende Hierarchie von Zertifizierungsstellen spezifiziert:

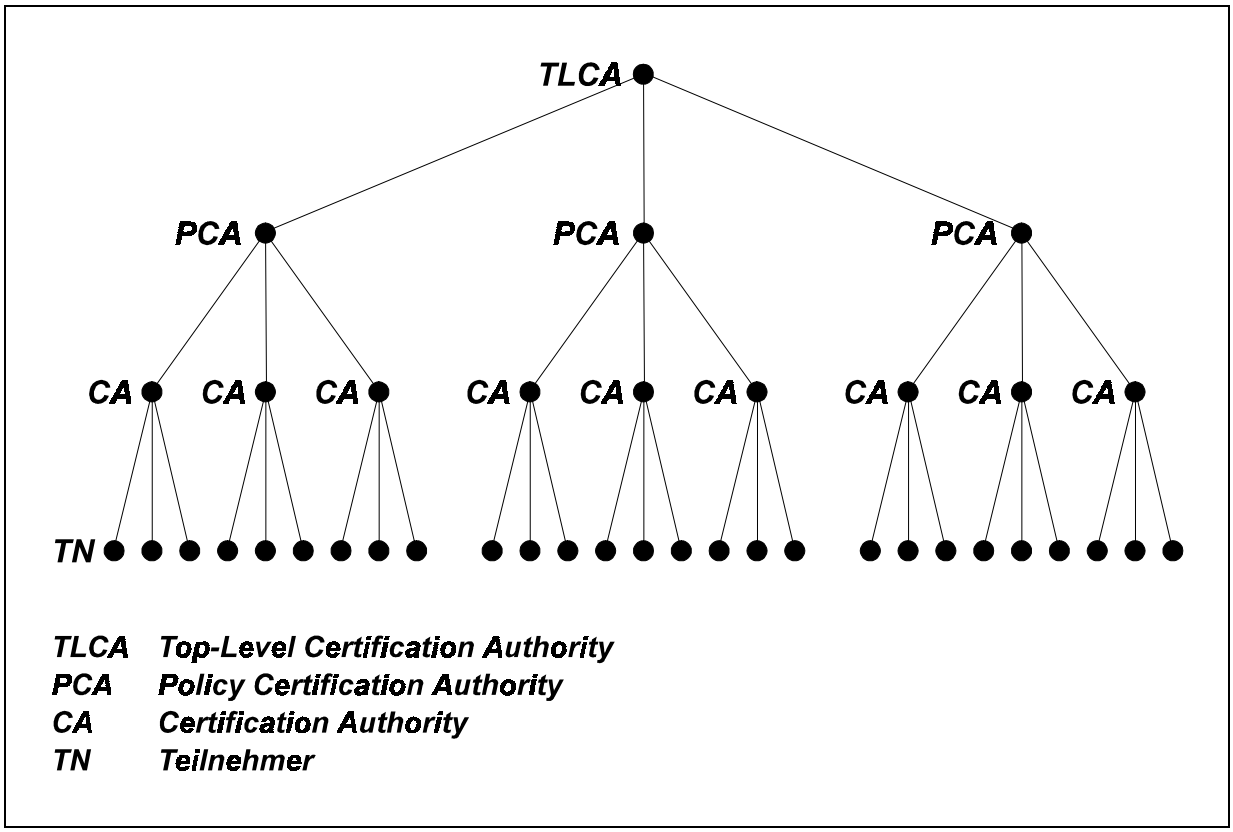

Bild 1: Zertifizierungshierarchie

Zu dieser Zertifizierungshierarchie gehören die folgenden Instanzen:

TLCA: Die Top-Level CA (TLCA) dient als Wurzel der Zertifizierungshierarchie. Ihr öffentlicher Schlüssel wird allgemein verfügbar gemacht und dient als Basis für den Schlüsselaustausch und damit für die Interoperabilität innerhalb der gesamten Infrastruktur.

Die TLCA stellt Zertifikate für PCAs aus und läßt diese damit für den Betrieb innerhalb der Zertifizierungsinfrastruktur zu.

PCA: Eine Policy CA (PCA) ist zuständig für die Organisation und Kontrolle des Betriebs in dem gesamten ihr nachgeordneten Teilbaum. Sie erläßt eine Security Policy, die den Umgang mit Vorgängen und Komponenten innerhalb ihres Teilbaums regelt und die für den gesamten nachgeordneten Teilbaum verbindlich ist.

Die PCA zertifiziert CAs und läßt diese damit für den Betrieb innerhalb des Teilbaums und unter den Regeln der von ihr herausgegeben Security Policy zu.

CA: Die Certification Authorities (CAs) sind dafür zuständig, die eigentlichen Teilnehmerzertifikate auszustellen und Teilnehmer damit für die Teilnahme innerhalb der Zertifizierungshierarchie zuzulassen.

CAs können auch weitere nachgeordnete CAs zertifizieren, so daß die CAs selbst wiederum mehrstufig gestaffelt sein können. Es wird jedoch empfohlen, aus Performance- und Übersichtlichkeitsgründen die gesamte Zertifizierungshierarchie so flach wie möglich zu halten.

TN: Die Teilnehmer sind die eigentlichen Nutzer der Zertifizierungsinfrastruktur. Sie haben im Rahmen der Zertifizierungsinfrastruktur keine weiteren Aufgaben und nutzen die innerhalb dieser Infrastruktur verteilten Zertifikate und Schlüsselinformationen zum Schlüsselaustausch und zur sicheren Kommunikation mit anderen Teilnehmern.

Die hier definierte Hierarchie kann je nach Bedarf systemspezifisch gekürzt werden. In einer konkreten Umsetzung kann eine solche Hierarchie statt auf der Ebene der TLCA auch auf PCA- oder CA-Ebene beginnen.

Wichtig dabei ist, daß

- die Hierarchie stets mit einer für alle Teilnehmer gemeinsamen Wurzel beginnt.
- die Rollen der hier definierten Instanzen beibehalten werden, gegebenenfalls ergänzt um Aufgaben, die durch den Wegfall höherer Hierarchieebenen zusätzlich übernommen werden müssen.
- die Hierarchie stets so aufgebaut ist, daß zu jedem Teilnehmer ein eindeutiger Pfad von der Wurzel der Hierarchie bis zu ihm selbst definiert ist.

In einer solchen Zertifizierungshierarchie genügt es einem Teilnehmer, den öffentlichen Schlüssel der Wurzel der Hierarchie (kurz: Wurzelschlüssel) sicher zu kennen, um mit Hilfe von Zertifikaten, die im Netz verteilt oder mit Nachrichten mitgeschickt werden, jeden anderen öffentlichen Schlüssel eines beliebigen Teilnehmer sicher erhalten und überprüfen zu können. Eine direkte Kontaktaufnahme zwischen Teilnehmern vor Aufnahme einer sicheren Kommunikation ist dabei nicht erforderlich.

Um einen beliebigen öffentlichen Schlüssel erhalten und überprüfen zu können, benötigt ein Teilnehmer den zu diesem Schlüssel gehörenden **Zertifizierungspfad**. Der Zertifizierungspfad besteht aus der Folge aller Zertifikate von dem fraglichen Schlüssel bis zur Wurzel der Zertifizierungshierarchie. Für die Überprüfung eines Teilnehmerschlüssels in der oben aufgezeichneten Hierarchie würde dafür beispielsweise benötigt:

- das Zertifikat der CA über den öffentlichen Teilnehmerschlüssel
- das Zertifikat der PCA über den öffentlichen CA-Schlüssel
- das Zertifikat der TLCA über den öffentlichen PCA-Schlüssel

(In obigem Bild entspricht ein Zertifikat jeweils einer Verbindungslinie, d.h. dem Übergang von einer Instanz auf die vor- bzw. nachgeordnete)

Der Teilnehmer prüft dann mit Hilfe des ihm sicher bekannten Wurzelschlüssels das Zertifikat der TLCA über den PCA-Schlüssel und erhält so auf sichere Weise den öffentlichen Schlüssel der PCA. Mit diesem prüft er das Zertifikat der PCA über den CA-Schlüssel und erhält so auf sichere Weise den öffentlichen Schlüssel der CA. Mit diesem wiederum kann er

abschließend das Zertifikat des gesuchten Teilnehmers prüfen und so auf sichere Weise in den Besitz des zugehörigen öffentlichen Teilnehmerschlüssels gelangen.

Die zu überprüfenden Zertifizierungspfade werden umso kürzer, je flacher die Zertifizierungshierarchie ist. Die Überprüfung eines Zertifizierungspfades kann weiter abgekürzt werden, wenn beide Teilnehmer über eine gemeinsame, in der Hierarchie weiter unten liegende Teilbaumwurzel verfügen. Verfügt der prüfende Teilnehmer neben dem Wurzelschlüssel auf sichere Weise auch über alle öffentlichen Schlüssel der Zertifizierungsstellen auf seinem eigenen Zertifizierungspfad, so braucht er dann die Überprüfung jeweils nur ab der gemeinsamen Teilbaumwurzel durchzuführen. Da Kommunikationsbeziehungen in einer Hierarchie in der Regel überwiegend lokal, d.h. im Bereich derselben CA oder PCA bleiben, kann hierdurch je nach Situation eine deutliche Rationalisierung erreicht werden.

Neben den oben genannten Zertifizierungsstellen (TLCA, PCA, CA) können im Rahmen einer Sicherheitsinfrastruktur weitere Organisationseinheiten eingerichtet werden. Zu nennen sind hier insbesondere Registration Authorities (RA) und Naming Authorities (NA).

Registration Authorities entlasten eine oder mehrere CAs, indem sie für diese den gesamten Kontakt zu den Teilnehmern übernehmen und abwickeln (Antragsbearbeitung, Personenprüfung, Aushändigung von Sicherheitstoken, etc.).

Naming Authorities können eine Hierarchie weiter entlasten, indem sie die einheitliche und kollisionsfreie Namensgebung für alle Teilnehmer und Instanzen übernehmen.

Beide Typen von Authorities können im Rahmen einer Sicherheitsinfrastruktur als zusätzliche Einheiten vorgesehen und organisatorisch integriert werden. Sie haben jedoch keine Auswirkung auf die Zertifizierungsinfrastruktur und auf das Schlüsselmanagement und werden daher im Rahmen dieser Spezifikation nicht weiter behandelt.

4.2.2 Zertifikate

Für die vorliegende Version 1 der MailTrusT-Spezifikation wird die Verwendung von X.509v1-Zertifikaten festgeschrieben.

Ein X.509v1-Zertifikat hat folgende ASN.1-Struktur:

```
Certificate ::= SEQUENCE {
    tbsCertificate          TBSCertificate,
    signatureAlgorithm      AlgorithmIdentifier,
    signature               BIT STRING
}

TBSCertificate ::= SEQUENCE {
    version                 [0] Version DEFAULT v1,
    serialNumber            CertificateSerialNumber,
    signature               AlgorithmIdentifier,
    issuer                  Name,
    validity                Validity,
    subject                 Name,
    subjectPublicKeyInfo    SubjectPublicKeyInfo
}
```

Die einzelnen Elemente haben dabei folgende Struktur und Bedeutung:

version:

```
Version ::= INTEGER{v1(0)}
```

Gibt die Version des X.509-Zertifikats an. In MailTrusT sind gemäß dieser Spezifikation nur X.509v1-Zertifikate zulässig.

serialNumber:

```
CertificateSerialNumber ::= INTEGER
```

Die serialNumber dient dazu, ein Zertifikat eindeutig zu identifizieren.

Eine CA muß sicherstellen, daß jedes von ihr ausgegebene Zertifikat eine für diese CA eindeutige „serialNumber“ erhält. Da innerhalb einer Zertifizierungsinfrastruktur die CA-Namen ebenfalls eindeutig sind (siehe Kapitel 5 „Festlegung von Namensstrukturen und -formaten“), ist damit ein Zertifikat durch Angabe der ausstellenden CA (siehe Zertifikatsfeld „issuer“) und der „serialNumber“ infrastrukturweit eindeutig identifizierbar und bezeichenbar.

Diese Kombination aus CA-Name und Seriennummer wird beispielsweise auch zur eindeutigen Referenzierung von Zertifikaten innerhalb von Sperrlisten verwendet (siehe Kapitel 4.2.3 „Sperrlisten“)

signature:

```
AlgorithmIdentifier ::= SEQUENCE {
    algorithm               ALGORITHM.&id({SupportedAlgorithms}),
    parameters              ALGORITHM.&Type
({SupportedAlgorithms}{@algorithm}) OPTIONAL
}

SupportedAlgorithms ALGORITHM ::= {...}
ALGORITHM ::= TYPE-IDENTIFIER
```

Dieses Feld gibt an, welcher Algorithmus von der CA zum Signieren des Zertifikats verwendet wurde (siehe auch Kapitel 8 „Kryptoalgorithmen“).

issuer:

```
Name ::= CHOICE {distinguishedName RDNSequence}
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::= SET OF AttributeValueAssertion
AttributeValueAssertion ::= SEQUENCE {AttributeType, AttributeValue}
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY
```

Das „issuer“-Feld gibt den Herausgeber des Zertifikats, d.h. die ausstellende CA an. In X.509v1 und damit nach dieser Spezifikation auch innerhalb von MailTrusT sind nur X.500-Distinguished-Names als Namen zugelassen.

validity:

```
Validity ::= SEQUENCE {
    notBefore               UTCTime,
    notAfter                UTCTime
}
```

Das Feld „validity“ gibt an, innerhalb welchen Zeitraums das Zertifikat gültig ist. „notBefore“ gibt den Anfang und „notAfter“ das Ende des Gültigkeitszeitraums an, wobei beide Zeitpunkte in den Gültigkeitszeitraum jeweils eingeschlossen sind.

In Anlehnung an die IETF-PKIX-Gruppe (siehe [PKIX1], Kapitel 4.1.5) wird empfohlen, Zeitpunkte stets in Greenwich Mean Time (GMT) anzugeben und die Angabe von Sekunden dabei nicht zu verwenden. Eine Zeitangabe hat dann das Format YYMMDDHHMMZ[8].

Applikationen müssen jedoch aus Kompatibilitätsgründen auch in der Lage sein, beliebige UTCTime-Formate (YYMMDDHHMMSS±HHMM) auszuwerten.

Da die Jahrtausendwende nicht allzu weit entfernt ist, wird für die Bedeutung der Jahreszahl in den Feldern „YY“ der obigen Formate folgendes festgelegt:

- Ein „YY“-Wert zwischen 65 und 99 bezieht sich auf das 20. Jahrhundert und verweist auf das entsprechende Jahr zwischen 1965 und 1999.
- Ein „YY“-Wert zwischen 00 und 64 bezieht sich auf das 21. Jahrhundert und verweist auf das entsprechende Jahr zwischen 2000 und 2064.

Formal ausgedrückt:

$65 \leq YY \leq 99 \quad \rightarrow \quad$ Jahr := 19YY
$00 \leq YY \leq 64 \quad \rightarrow \quad$ Jahr := 20YY

subject:

Das „subject“-Feld gibt an, für wen das Zertifikat ausgestellt wurde, wer also Inhaber des Zertifikats und damit auch des darin enthaltenen öffentlichen und des zugehörigen privaten Schlüssels ist.

Das „subject“-Feld hat dasselbe Format wie das oben bereits beschriebene „issuer“-Feld. Auch für das subject-Feld sind im Rahmen dieser Spezifikation nur X.500-Distinguished -Names zulässig.

Die „subject“-Namen müssen innerhalb einer Infrastruktur eindeutig sein. Kapitel 5 „Festlegung von Namensstrukturen und -formaten“ regelt, wie diese Eindeutigkeit in MailTrusT erreicht wird.

---

[8] Das ‘Z’ am Ende steht für „Zulu-Zeit“ und ist die allgemein übliche Kurzform für GMT.

subjectPublicKeyInfo:

```
SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm              AlgorithmIdentifier,
    subjectPublicKey       BIT STRING
}
```

Dieses Feld enthält den durch das Zertifikat zertifizierten öffentlichen Schlüssel des Zertifikatsinhabers (siehe Feld „subject“).

„subjectPublicKey“ enthält den zertifizierten öffentlichen Schlüssel.
„algorithm“ gibt an, mit welchem Kryptoalgorithmus dieser Schlüssel zu verwenden ist.

Kapitel 8 „Kryptoalgorithmen“ nennt die in MailTrusT verfügbaren Algorithmenidentifier und gibt darüber hinaus an, für welchen Zweck (Signieren, Verschlüsseln) der jeweilige Algorithmus verwendet werden darf.

4.2.3 Sperrlisten

Zum Sperren von Zertifikaten vor Ablauf ihrer ursprünglichen Gültigkeitsdauer sind in MailTrusT Sperrlisten gemäß [RFC1422] vorgesehen.

Eine Sperrliste (Certificate Revocation List - CRL) hat folgende ASN.1-Struktur:

```
CertificateRevocationList ::= SIGNED SEQUENCE {
    signature              AlgorithmIdentifier,
    issuer                 Name,
    lastUpdate             UTCTime,
    nextUpdate             UTCTime,
    revokedCertificates    SEQUENCE OF CRLEntry OPTIONAL
}
```

signature:

```
AlgorithmIdentifier ::= SEQUENCE {
    algorithm              ALGORITHM.&id({SupportedAlgorithms}),
    parameters             ALGORITHM.&Type
({SupportedAlgorithms}{@algorithm}) OPTIONAL
}

SupportedAlgorithms ALGORITHM ::= {...}
ALGORITHM ::= TYPE-IDENTIFIER
```

Dieses Feld gibt an, welcher Algorithmus von der CA zum Signieren der Sperrliste verwendet wurde (siehe auch Kapitel 8 „Kryptoalgorithmen“).

issuer:

```
Name ::= CHOICE {distinguishedName RDNSequence}
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::= SET OF AttributeValueAssertion
AttributeValueAssertion ::= SEQUENCE {AttributeType, AttributeValue}
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY
```

Das issuer-Feld gibt den Herausgeber der Sperrliste an. Das ist jeweils die Zertifizierungsstelle, die die in der Sperrliste referenzierten Zertifikate ursprünglich generiert hat. Der Name des Herausgebers der Sperrliste ist in Form eines X.500-Distinguished-Name anzugeben.

lastUpdate:

Gibt den Zeitpunkt an, zu dem diese Sperrliste herausgegeben wurde.

Wie bei Zertifikaten (siehe 4.2.2) wird im Rahmen dieser Spezifikation empfohlen, Zeitpunkte stets in Greenwich Mean Time (GMT) anzugeben und die Angabe von Sekunden dabei nicht zu verwenden, d.h. die Zeit insgesamt im Format YYMMDDHHMMZ anzugeben. Applikationen müssen jedoch aus Kompatibilitätsgründen auch in der Lage sein, beliebige UTCTime-Formate auszuwerten.

nextUpdate:

Gibt den Zeitpunkt an, zu dem die nächste Sperrliste der betreffenden CA herausgegeben werden wird. Anhand dieser Angabe kann entschieden werden, ob die Sperrliste noch aktuell ist.

revokedCertificates:

```
SEQUENCE OF CRLEntry

CRLEntry ::= SEQUENCE {
    userCertificate         SerialNumber,
    revocationDate          UTCTime
}

SerialNumber ::= INTEGER
```

„revokedCertificates“ ist die Liste der gesperrten Zertifikate. Die Zertifikate werden in diese Liste nicht vollständig, sondern nur als Referenz aufgenommen. Die Zertifikate werden referenziert durch die Seriennummer in „userCertificate“ und den CA-Namen im Sperrlistenfeld „issuer“. Diese Angaben identifizieren ein Zertifikat eindeutig innerhalb der gesamten Sicherheitsinfrastruktur (siehe 4.2.2 „Zertifikate“, Feld „serialNumber“).

4.2.4 Gültigkeit von Schlüsseln und Zertifikaten

Die Gültigkeit von Schlüsseln und Zertifikaten wird gemäß [RFC1422] wie folgt definiert:

- Ein **Schlüssel(-paar)** ist zu einem bestimmten Zeitpunkt genau dann gültig, wenn zu diesem Zeitpunkt der zugehörige Zertifizierungspfad gültig ist.
- Ein **Zertifizierungspfad** (siehe 4.2.1 „Schlüsselmanagement“) ist zu einem bestimmten Zeitpunkt genau dann gültig, wenn alle in ihm enthaltenen Zertifikate zu diesem Zeitpunkt gültig sind.
- Ein **Zertifikat** ist zu einem bestimmten Zeitpunkt genau dann gültig, wenn
  - die Signatur des Zertifikats gültig ist.
  - der fragliche Zeitpunkt innerhalb des Gültigkeitszeitraums des Zertifikats liegt (also zwischen „notBefore“ und „notAfter“, jeweils einschließlich).
  - das Zertifikat nicht in der zum Zeitpunkt der Prüfung (!) aktuellen Sperrliste der ausstellenden Zertifizierungsstelle enthalten ist oder das Zertifikat in dieser Sperrliste enthalten ist, der angegebene Sperrzeitpunkt jedoch nach dem fraglichen Zeitpunkt liegt.

4.2.5 Online-Dienste

Die MailTrust-Spezifikation geht in der vorliegenden Version nicht von der Verfügbarkeit und Nutzung eines X.500-Verzeichnisses aus. Eine Integration von X.500-Diensten wird im Rahmen dieser Spezifikation nicht gefordert.

Gemäß [RFC1422] und [RFC1424] werden für die Kommunikation zwischen Teilnehmern und ihren zugehörigen CAs und entsprechend zwischen CAs, PCAs und TLCA folgende zusätzlichen Austauschformate spezifiziert:

- Zertifizierungsanfrage zur Zertifizierung eines durch den Teilnehmer generierten Schlüssels durch die zuständige CA
- zugehörige Zertifizierungsantwort
- Abruf einer Sperrliste
- Übermittlung einer Sperrliste

Im Rahmen dieser Spezifikation ist zur Zeit nicht definiert, wie Zertifikate automatisch von einer zentralen Stelle (Zertifizierungsstelle oder Verzeichnisdienst) angefordert werden können. Die Zertifikate selbst sind im Rahmen eines solchen Dienstes stets in einem der in Kapitel 3 spezifizierten Nachrichtenformate auszuliefern.

Typischerweise werden Zertifikate jeweils in den PEM- bzw. MTT-Nachrichten mitgeliefert oder sind bei Bedarf als PEM- oder MTT-Nachricht von dem gewünschten Teilnehmer anzufordern.

4.2.5.1 Zertifizierungsanfrage

Eine Zertifizierungsanfrage ist eine „MIC-CLEAR“- oder „MIC-ONLY“- Nachricht, die als „Originator-Certificate“ ein vom Teilnehmer selbst unterschriebenes Zertifikat über seinen eigenen, zur Zertifizierung vorgelegten öffentlichen Schlüssel enthält („self-signed certificate“).

Die Felder dieses selbst unterschriebenen Zertifikats sind wie folgt zu belegen:

version	0 (X.509v1-Zertifikat)
serialNumber	beliebig, Vorschlag: 0
signature	Algorithmus, mit dem der Teilnehmer das selbst erzeugte Zertifikat signiert hat.
issuer	X.500-Distinguished-Name des Teilnehmers
validity	beliebig, Vorschlag: notBefore = notAfter = 1. Januar 1970, 12:00
subject	X.500-Distinguished-Name des Teilnehmers (wie „issuer“)
subjectPublicKeyInfo	zu zertifizierender öffentlicher Schlüssel des Teilnehmers

Der in „signature“ angegebene Algorithmus muß asymmetrisch sein und eine Hashfunktion beinhalten, zu deren Berechnung kein kryptographischer Schlüssel erforderlich ist, damit das Zertifikat von Externen ohne weitere Informationen oder Schlüssel geprüft werden kann.

4.2.5.2 Zertifizierungsantwort

Die Zertifizierungsantwort entsteht aus der Zertifizierungsanfrage dadurch, daß das in der Anfrage noch vom Teilnehmer selbst unterschriebene Zertifikat durch das neue Zertifikat der CA ersetzt wird und gegebenenfalls weitere, zum Zertifizierungspfad des Teilnehmers gehörende „Issuer-Certificate“-Felder eingefügt werden. Nachrichteninhalt und Signatur der Nachricht bleiben unverändert.

4.2.5.3 Abruf einer Sperrliste

Zum Abruf einer Sperrliste wird ein neuer Nachrichtentyp „Proc-Type: 4,CRL-RETRIEVAL-REQUEST“ definiert.

Ein „CRL-RETRIEVAL-REQUEST“ enthält das angegebene Proc-Type-Feld und ein oder mehrere „Issuer:“-Felder. Jedes „Issuer:“-Feld enthält den X.500-Namen einer Zertifizierungsinstanz, deren Sperrliste geliefert werden soll, kodiert nach ASN.1 BER und dem in 3.2.3.4 spezifizierten Verfahren.

Beispiel eines CRL-Abrufs:

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Proc-Type: 4,CRL-RETRIEVAL-REQUEST
Issuer: <X.500-Name der ersten CA>
Issuer: <X.500-Name der zweiten CA>
-----END PRIVACY-ENHANCED MESSAGE-----
```

4.2.5.4 Übermittlung einer Sperrliste

Zur Übermittlung einer Sperrliste wird ebenfalls ein neuer Nachrichtentyp definiert.

Eine Sperrlisten-Nachricht beginnt mit der Kopfzeile „Proc-Type: 4,CRL“.

Danach folgen ein oder mehrere „CRL:“-Zeilen, in denen jeweils eine Sperrliste eingetragen ist. Die einzelnen Sperrlisten sind nach ASN.1 BER und dem in 3.2.3.4 spezifizierten Verfahren zu kodieren und anschließend nach dem in 3.2.4 spezifizierten Verfahren umzubrechen.

Jeder „CRL:“-Eintrag kann von einem „Originator-Certificate:“ und gegebenenfalls einem oder mehreren „Issuer-Certificate:“-Feldern, wie in 3.2.4.6 bzw. 3.2.4.8 spezifiziert, gefolgt werden, die den Zertifizierungspfad des Herausgebers der Sperrliste definieren.

Beispiel für eine „CRL“-Nachricht:

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Proc-Type: 4,CRL
CRL: <Sperrliste der ersten CA>
Originator-Certificate: <Zertifizierungspfad der ersten CA>
Issuer-Certificate:
CRL: <Sperrliste der zweiten CA>
Originator-Certificate: < Zertifizierungspfad der zweiten CA>
Issuer-Certificate:
-----END PRIVACY-ENHANCED MESSAGE-----
```

5 Festlegung von Namensstrukturen und -formaten

Aufgrund der Verwendung von X.509v1-Zertifikaten sind als Namen von Teilnehmern und Zertifizierungsstellen gemäß dieser Spezifikation stets X.500-Distinguished-Names zu verwenden.

Im Rahmen einer Sicherheitsinfratruktur nach dieser Spezifikation ist darüber hinaus durch geeignete Regelungen sicherzustellen, daß alle innerhalb der Infrastruktur vergebenen Namen infrastrukturweit eindeutigt sind. Es darf in der gesamten Infrastruktur weder zwei Teilnehmer mit gleichem Namen, noch zwei Zertifizierungsstellen mit gleichem Namen, noch einen Teilnehmer und eine Zertifizierungsstelle mit gleichem Namen geben können.

Werden innerhalb einer Sicherheitsinfrastruktur keine speziellen Regelungen getroffen, um die Eindeutigkeit von Namen sicherzustellen, so sind die in PEM definierten Regeln der Namenssubordination anzuwenden, mit Hilfe derer ebenfalls eine eindeutige Vergabe von Namen erreicht wird:

- Unterhalb der CA-Ebene gilt die Regel der Namenssubordination: CAs dürfen nur Zertifikate ausstellen, bei denen der im „subject“-Feld eingetragene Subjektname ihren eigenen CA-Namen als Präfix enthält.
- CAs dürfen innerhalb ihres Bereichs nur eindeutige Namen vergeben.
- Die Wurzel einer Zertifizierungshierarchie (in der Regel die TLCA oder eine PCA) hat eine Datenbank zu führen, in der die Namen aller PCAs und CAs geführt und bei Neuzulassung weiterer CAs und PCAs auf Kollisionsfreiheit geprüft werden.
- Die Namen von CAs, die wiederum selbst von einer CA zertifiziert wurden (mehrstufige CA-Hierarchie, siehe 4.2.1) können bei Bedarf aus der Datenbank weggelassen werden, da ihre Namenskollisionsfreiheit bereits durch die Namenssubordinationsregel für die übergeordneten CAs gewährleistet ist.
- Die Wurzel der Zertifizierungshierarchie hat die Aufgabe, die Kollisionsfreiheit aller Namen zu garantieren, die für CAs, PCAs und für sie selbst vergeben werden. Jede CA hat die Aufgabe, die Eindeutigkeit der von ihr zertifizierten Namen zu garantieren. In Zusammenwirken mit der Namenssubordination wird damit insgesamt die Kollisionsfreiheit aller Namen innerhalb der gesamten Sicherheitsinfrastruktur gewährleistet.

6 Spezifikation der Funktionalität und des Formats der PSE

In diesem Abschnitt wird eine Schnittstelle zwischen PSE und Applikation spezifiziert. Ziel dieser Spezifikation ist es, Applikationen und PSE-Module klar zu trennen und damit

- eine getrennte Entwicklung von Applikationen und PSE-Modulen zu ermöglichen
- es einem Benutzer zu ermöglichen, seine PSE für beliebige MTT-kompatible Applikationen zu nutzen
- es den CA zu ermöglichen, PSEs ohne Kenntnis und unabhängig von den eingesetzten Applikationen zu generieren und zu verteilen

Als Schnittstelle zwischen Applikation und PSE wird hiermit die in [PKCS11] von RSA Inc. spezifizierte „Cryptoki“-Schnittstelle festgelegt.

Folgende Elemente der „Cryptoki“-Schnittstelle werden im Rahmen der MailTrusT-Spezifikation nicht benötigt und können daher bei Realisierungen ausgelassen werden:

Folgende ***Funktionen*** müssen **nicht** realisiert werden:

- C_SignRecoverInit
- C_SignRecover
- C_VerifyRecoverInit
- C_VerifyRecover
- C_DeriveKey

Folgende ***Mechanismen*** müssen **nicht** realisiert werden:

- CKM_RSA_9796
- CKM_DSA_KEY_PAIR_GEN
- CKM_DSA
- CKM_DH_PKCS_KEY_PAIR_GEN
- CKM_DH_PKCS_DERIVE
- CKM_RC2_KEY_GEN
- CKM_RC2_ECB
- CKM_RC2_CBC
- CKM_RC2_MAC
- CKM_RC4_KEY_GEN
- CKM_RC4
- CKM_DES_MAC
- CKM_DES2_KEY_GEN
- CKM_DES3_MAC

7 Festlegung der Kodierung verschiedener Dokumentenformate

Durch Einführung der neuen Nachrichtentypen „MTT-1,MIC-BIN“, „MTT-1,ENCRYPTED-BIN“, „MTT-1,MIC-RAW“ und „MTT-1,ENCRYPTED-RAW“ (siehe 3.2.1) können nach dieser Spezifikation beliebige Datenformate ohne Vorab-Kodierung direkt bearbeitet werden. Eine spezielle Kodierung für Nicht-Text-Dokumente entfällt damit.

Im Kopffeld „Content-Domain“ (siehe 3.2.4.2) kann für MTT-Nachrichtentypen im Subfeld <data-type> angegeben werden, in welchem Datenformat die Original-Daten vorliegen.

Aufgrund der speziellen Anforderungen aus dem MVS-Bereich wird dabei zwischen MVS-Datenformaten und allgemeinen Datenformaten unterschieden. MVS-Datenformate sind durch den Präfix „MVS-“ gekennzeichnet und so von allgemeinen Datenformaten unterscheidbar.

7.1 Allgemeine Datenformate

Die nachfolgende Tabelle enthält die im Rahmen dieser Spezifikation definierten Formatbezeichner für allgemeine Datenformate. Andere Bezeichner, die nicht in dieser Tabelle enthalten sind, sind zulässig und jeweils geeignet abzustimmen.

Folgende Werte sind für allgemeine Datenformate definiert:

Wert	Bedeutung
Adobe-ILL	Adobe Illustrator
AmiPro	Ami Professional
AutoCAD	AutoCAD DXF
Binary	Allgemeine Binärdaten
BMP	Windows/OS-2 Bitmap
CGM	Computer Graphics Metafile
CorelCRT	CorelCHART
CorelDRW	CorelDRAW
CorelEXC	Corel Presentation Exchange
CorelPHT	CorelPHOTO-PAINT
Draw	Micrographx Draw
DVI	Device Independent Format, DVI-Datei
EPS	(Placable) Encapsulated Postscript
Excel	Excel-Tabelle
GEM	GEM-Dateien
GIF	Compuserve Bitmaps, „GIF“-Datei
HPGL	HPGL Plotter Datei
JPEG	JPEG Bitmap

Wert	Bedeutung
Kodak	Kodak Photo-CD
LaTeX	LaTeX-Datei
Lotus	Lotus 123 1A
Lotus-PIC	Lotus PIC
Mac-PICT	Macintosh PICT
Mac-Word	Microsoft Word für Macintosh
MS-WfD	Microsoft Word für DOS
MS-Word MS-Word-2 MS-Word-6	Microsoft Word für Windows Microsoft Word für Windows, Version 2.0 Microsoft Word, Version 6 und 7
PIF	IBM Program Information File
Postscript	(Interpreted) Postscript
RTF	Microsoft Rich Text Format
SCITEX	SCITEX
TAR	UNIX Tape ARchive
Targa	TARGA Bitmap
TeX	Plain-TeX Datei
Text	ASCII-Text-Datei
TIFF	TIFF-Bitmap
TIFF-FC	TIFF Four-Color Datei
UID	User Interface Database, UID-Datei (OSF/Motif)
UUEncode	UUEncode-te Datei
WMF	Windows Metafile
WordPerfect	WordPerfect
WP-Grph	WordPerfect Graphic

Tabelle 5: Allgemeine Datenformate

7.2 MVS-Datenformate

Für MVS-Formate wird das Subfeld <data-type> wie folgt weiter unterteilt:

<data-type> ::= <recfm>/<lrecl>[/blksize>]

<lrecl> gibt dabei die logische Recordlänge an, <blksize> die Blockgröße.

Die Angabe von <blksize> ist optional. Wird sie weggelassen, so entfällt auch der direkt davor stehende Schrägstrich.

<recfm> gibt das Record-Format an. Folgende Record-Formate sind definiert:

Kennung	Format
MVS-F	Fixed-length, unblocked
MVS-FA	Fixed-length, ASA print-control characters
MVS-FB	Fixed-length, blocked
MVS-FM	Fixed-length, machine print-control characters
MVS-FS	Fixed-length, unblocked, standard
MVS-FBA	Fixed-length, blocked, ASA print-control characters
MVS-FBM	Fixed-length, blocked, machine print-control characters
MVS-FBS	Fixed-length, blocked, standard
MVS-FSA	Fixed-length, unblocked, standard, ASA print-control characters
MVS-FSM	Fixed-length, unblocked, standard, machine print-control characters
MVS-FBSM	Fixed-length, blocked, standard, machine print-control characters
MVS-FBSA	Fixed-length, blocked, standard, ASA print-control characters
MVS-V	Variable-length, unblocked
MVS-VA	Variable-length, ASA print-control characters
MVS-VB	Variable-length, blocked
MVS-VM	Variable-length, machine print-control characters
MVS-VS	Variable-length, unblocked, standard
MVS-VBA	Variable-length, blocked, ASA print-control characters
MVS-VBM	Variable-length, blocked, machine print-control characters
MVS-VBS	Variable-length, blocked, standard
MVS-VSA	Variable-length, unblocked, standard, ASA print-control characters
MVS-VSM	Variable-length, unblocked, standard, machine print-control characters
MVS-VBSM	Variable-length, blocked, standard, machine print-control characters
MVS-VBSA	Variable-length, blocked, standard, ASA print-control characters
MVS-U	Undefined-length
MVS-UA	Undefined-length, ASA print-control characters
MVS-UM	Undefined-length, machine print-control characters

Tabelle 6: MVS-Datenformate

Beispiel:

Ein MVS-Datenformat mit den Parametern „fixed-length, blocked“, „Recordlänge: 80“ und „Blocklänge: 800“ wird beispielsweise kodiert als:

<data-type> = MVS-FB/80/800

8 Kryptoalgorithmen

In diesem Abschnitt sind die gemäß dieser Spezifikation verwendbaren Kryptoalgorithmen aufgelistet. Die Algorithmen sind dabei so in Klassen zusammengefasst, wie sie im Laufe der Transformationsschritte (siehe 3.2.3) benötigt werden.

Eine Applikation, die nach dieser Spezifikation arbeitet, muß in der Lage sein, bei ankommenden Nachrichten alle hier spezifizierten Algorithmen zu erkennen und zu verarbeiten. Bei abgehenden Nachrichten ist die Auswahl unter den hier spezifizierten Algorithmen freigestellt.

Bei allen hier spezifizierten Algorithmen werden, soweit nicht explizit anders angegeben, Ein- und Ausgabedaten folgenderweise dargestellt und übergeben: Bitfolgen (Schlüssel, Hashwerte, Nachrichten etc.) werden stets als eine Folge von Bytes dargestellt. Das höchstwertige Byte einer Bytefolge wird bei serieller Übergabe als erstes übergeben und in textueller Darstellung (z.B. in dieser Beschreibung oder in PEM-/MTT-Nachrichten) an der ersten, d.h. am weitesten links befindlichen Position angegeben. Das niederwertigste Byte einer Bytefolge wird entsprechend als letztes übertragen bzw. an der am weitesten rechts befindlichen Position angegeben. Alle anderen Bytes stehen in ihrer natürlichen Reihenfolge zwischen dem höchst- und dem niederwertigsten Byte.

Für jeden der hier spezifizierten Algorithmen ist in den Einzelbeschreibungen angegeben, für welchen Verwendungszweck er gedacht ist (siehe auch Kapitel 4 „Spezifikation der Zertifizierungsinfrastruktur“, insbesondere Kapitel 4.1, vorletzter und letzter Absatz).

Folgende Verwendungszwecke werden unterschieden:

- Verschlüsseln („encrypt“)
- Signieren von Daten („sign_data“)
- Signieren von Zertifikaten und Sperrlisten („sign_cert“)

Damit ist es einerseits möglich, Signatur- und Verschlüsselungsschlüssel voneinander zu trennen. Andererseits kann durch Ausgabe entsprechender Zertifikate damit auch festgelegt werden, welche Instanzen selbst wiederum Zertifikate ausstellen dürfen.

8.1 Hashen von Daten

Zum Berechnen eines Hashwerts in Vorbereitung einer digitalen Signatur in Transformationsschritt „Signieren“ (siehe 3.2.3.2) können folgende Algorithmen eingesetzt werden:

8.1.1 MD2

- **Objekt-Identifier**:
  { iso(1) member-body(2) US(840) rsadsi(113549) digestAlgorithm(2) md2(2) }
- **Bezeichnung in Kopfzeilen**: "RSA-MD2"
- **ASN.1-Parameter im Typ „AlgorithmIdentifier“**: NULL
- **Verwendungszweck** (siehe Kapitel 8): sign_data
- **Referenz und Kurzbeschreibung**:

  Der MD2-Algorithmus ist in [RFC1319] spezifiziert.

  Gemäß [RFC1423] ist in der Spezifikation des MD2 in RFC1319 ein Fehler enthalten. In Abschnitt 3.2 ist der Text "`Set C[j] to S[c xor L]`" zu ersetzen durch "`Set C[j] to S[c xor L] xor C[j]`".

Der MD2 nimmt als Eingabe eine Nachricht beliebiger Länge und liefert als Ergebnis einen 16-Byte-Wert.

Bei Verwendung mit einem symmetrischen Schlüsselmanagement wird das Ergebnis des MD2 in zwei 8-Byte-Hälften aufgeteilt[9], die jeweils einzelnen mit dem symmetrischen Verschlüsselungsalgorithmus verschlüsselt und anschließend wieder zu einer 16-Byte-Kette zusammengesetzt werden. Dieser Wert wird hexadezimal kodiert als viertes Argument in dem entsprechenden „Key-Info“-Feld (siehe 3.2.4.11) eingetragen.

8.1.2 MD5

- **Objekt-Identifier**:
  { iso(1) member-body(2) US(840) rsadsi(113549) digestAlgorithm(2) md5(5) }
- **Bezeichnung in Kopfzeilen**: "RSA-MD5"
- **ASN.1-Parameter im Typ „AlgorithmIdentifier“**: NULL
- **Verwendungszweck** (siehe Kapitel 8): sign_data
- **Referenz und Kurzbeschreibung**:

  Der MD5-Algorithmus ist in [RFC1321] spezifiziert.

  Der MD5 nimmt als Eingabe eine Nachricht beliebiger Länge und liefert als Ergebnis einen 16-Byte-Wert.

  Bei Verwendung mit einem symmetrischen Schlüsselmanagements wird das Ergebnis des MD5 in zwei 8-Byte-Hälften aufgeteilt (siehe Fußnote zu MD2), die jeweils einzelnen mit dem symmetrischen Verschlüsselungsalgorithmus verschlüsselt und anschließend wieder zu einer 16-Byte-Kette zusammengesetzt werden. Dieser Wert wird hexadezimal kodiert als viertes Argument in dem entsprechenden „Key-Info“-Feld (siehe 3.2.4.11) eingetragen.

8.1.3 SHA-1

- **Objekt-Identifier**:
  { iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) sha-1(26) }
- **Bezeichnung in Kopfzeilen**: "SHA-1"
- **ASN.1-Parameter im Typ „AlgorithmIdentifier“**: NULL
- **Verwendungszweck** (siehe Kapitel 8): sign_data
- **Referenz und Kurzbeschreibung**:

  Der SHA-1 ist in [FIPS180-1] spezifiziert.

  Der SHA-1 nimmt als Eingabe eine Nachricht beliebiger Länge und liefert als Ergebnis einen 20-Byte-Wert.

  Bei Verwendung mit einem symmetrischen Schlüsselmanagements wird das Ergebnis des SHA-1 zunächst auf 24 Byte verlängert (siehe Fußnote zu MD2), indem dem 20-Byte-Ergebnis des SHA-1 vier weitere beliebige Bytes als höchstwertige Bytes vorangestellt werden. Die so entstehenden 24 Byte werden in drei 8-Byte-Gruppen aufgeteilt, die jeweils einzelnen mit dem symmetrischen Verschlüsselungsalgorithmus verschlüsselt und anschließend wieder zu einer 24-Byte-Kette zusammengesetzt

---

9 Zur Zeit sind im Rahmen dieser Spezifikation für symmetrisches Schlüsselmanagement nur Algorithmen vorgesehen, die auf 8-Byte-Blöcken arbeiten.

werden. Dieser Wert wird hexadezimal kodiert als viertes Argument in dem entsprechenden „Key-Info“-Feld (siehe 3.2.4.11) eingetragen.

8.2 Verschlüsseln von Daten

8.2.1 DES im CBC-Mode

- **Objekt-Identifier**:
  { iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) des-cbc(7) }
- **Bezeichnung in Kopfzeilen**: "DES-CBC"
- **ASN.1-Parameter im Typ „AlgorithmIdentifier“**: CBCParameter
  Als Parameter ist hier der Initialisierungsvektor (IV) für den CBC-Modus einzutragen, der im Gegensatz zum Schlüssel nicht geheimgehalten werden muß.
- **Verwendungszweck** (siehe Kapitel 8): encrypt
- **Referenz und Kurzbeschreibung**:

  Der DES (Data Encryption Standard) ist standardisiert in [FIPS46] und in [X3.92]. Der Verschlüsselungsmodus CBC (Cipher Block Chaining) ist standardisiert in [FIPS81], [X3.106] und [ISO10116].

  Vor Verschlüsselung mit DES-CBC müssen die Eingabedaten so aufgefüllt werden, daß die Länge der Eingabedaten in Bytes ein Vielfaches von 8 ist (Padding). Hierzu werden an die Eingabedaten 1 bis 8 Byte angehängt. Sei n die Länge der Eingabedaten in Byte. Dann werden 8-(n mod 8) Bytes mit dem Wert 8-(n mod8) angehängt.

  Folgende Sequenzen in hexadezimaler Darstellung sind damit als Padding möglich: 01, 0202, 030303, 04040404, 0505050505, 060606060606, 07070707070707, 0808080808080808. Dieses Padding kann nach Entschlüsselung der Daten wieder eindeutig entfernt werden.

  Für den DES werden Schlüssel mit 64 Bit Länge benötigt. Von diesen 64 Bit wird jedes 8. Bit als Paritätsbit verwendet, nur die verbleibenden 56 Bit werden tatsächlich als Schlüsselbits verwendet. Für jede Nachricht ist ein neuer Nachrichtenschlüssel zu erzeugen. Die Paritätsbits sind dabei stets auf ungerade Parität einzustellen. Beim Empfang einer Nachricht wird bei Verwendung eines symmetrischen Schlüsselmanagements empfohlen, die Paritätsbits zu prüfen, um mögliche Übertragungsfehler zu erkennen. Bei Verwendung eines asymmetrischen Schlüsselmanagements wird die Prüfung der Paritätsbits nicht empfohlen. Übertragungsfehler sind über das übergeordnete PKCS#1-Format erkennbar. Durch das Auslassen der Prüfung der Paritätsbits wird die Interoperabilität in den Fällen verbessert, in denen keine Übertragungsfehler vorlagen, der Sender jedoch entgegen der in [RFC1423] und in dieser Spezifikation getroffenen Festlegung keine korrekten Paritätsbits eingefügt hat.

  DES-Schlüssel werden, auch in der verschlüsselten Form, stets in voller Länge, d.h. mit allen 64 Schlüsselbits einschließlich Paritätsbits übertragen.

  Für den DES ist im CBC-Mode darüber hinaus ein 64-Bit langer Initialisierungsvektor (IV) erforderlich. Für jede zu verschlüsselnde Nachricht ist ein neuer Initialisierungsvektor zu generieren.

  Beim Erzeugen verschlüsselter Nachrichten werden im „DEK-Info“-Feld zwei Argumente eingetragen. Das erste Argument lautet „DES-CBC“ und referenziert den hier spezifizierten Algorithmus. In dem zweiten Argument wird der zugehörige Initialisierungsvektor eingetragen, hexadezimal kodiert in 16 Hexadezimalziffern.

Bei symmetrischem Schlüsselmanagement wird ausserdem der verschlüsselte DES-Schlüssel in hexadezimaler Kodierung als drittes Element im „Key-Info“-Feld eingetragen.

8.2.2 Triple-DES im CBC-Mode

- **Objekt-Identifier**:
  { iso(1) identified-organization(3) teletrust(36) algorithm(3) encryption-algorithm(1) des3(3) cbc(2)}

  Achtung ! Im Rahmen von TeleTrusT wird der angegebene Objekt-Identifier typischerweise noch durch eine weitere Zahl ergänzt, die die Art des „Paddings“ (s.u.) angibt. Aus Zeitgründen war es leider nicht möglich, einen geeigneten Objekt-Identifier zu bestimmen, der gleichzeitig das unten angegebene Padding bezeichnet. Es ist daher durchaus möglich, daß der angegebene Objekt-Identifier in einer Folgeversion der Spezifikation durch einen anderen Objekt-Identifier ersetzt wird. Realisierungen sollten deshalb so gebaut sein, daß Objekt-Identifier ohne großen Aufwand durch andere Objekt-Identifier (mit gleicher Bedeutung!) ersetzt werden können.

- **Bezeichnung in Kopfzeilen**: "DES3-CBC"

- **ASN.1-Parameter im Typ „AlgorithmIdentifier“**: CBCParameter
  Als Parameter ist hier der Initialisierungsvektor (IV) für den CBC-Modus einzutragen, der im Gegensatz zum Schlüssel nicht geheimgehalten werden muß.

- **Verwendungszweck** (siehe Kapitel 8): encrypt

- **Referenz und Kurzbeschreibung**:

  Der Triple-DES ist in [X9.17] standardisiert. Für ihn wird im Gegensatz zum Standard DES ein Paar von 64-Bit-Schlüsseln benötigt, insgesamt also 128 Bit. Der Verschlüsselungsmodus CBC (Cipher Block Chaining) ist standardisiert in [FIPS81], [X3.106] und [ISO10116].

  Das „Padding“ der Daten vor Bearbeitung mit DES3-CBC ist identisch zu dem für DES-CBC angegebenen „Padding“ (siehe 8.2.1) und erfolgt auf folgende Weise:

  Vor Verschlüsselung mit DES3-CBC werden die Eingabedaten so aufgefüllt, daß die Länge der Eingabedaten in Bytes ein Vielfaches von 8 ist. Hierzu werden an die Eingabedaten 1 bis 8 Byte angehängt. Sei n die Länge der Eingabedaten in Byte. Dann werden 8-(n mod 8) Bytes mit dem Wert 8-(n mod8) angehängt.

  Folgende Sequenzen in hexadezimaler Darstellung sind damit als Padding möglich: 01, 0202, 030303, 04040404, 0505050505, 060606060606, 07070707070707, 0808080808080808. Dieses Padding kann nach Entschlüsselung der Daten wieder eindeutig entfernt werden.

  Für den Triple-DES werden Schlüssel mit 128 Bit Länge benötigt. Von diesen 128 Bit wird jedes 8. Bit als Paritätsbit verwendet, nur die verbleibenden 112 Bit werden tatsächlich als Schlüsselbits verwendet. Für jede Nachricht ist ein neuer Nachrichtenschlüssel zu erzeugen. Die Paritätsbits sind dabei stets auf ungerade Parität einzustellen. Beim Empfang einer Nachricht wird bei Verwendung eines symmetrischen Schlüsselmanagements empfohlen, die Paritätsbits zu prüfen, um mögliche Übertragungsfehler zu erkennen. Bei Verwendung eines asymmetrischen Schlüsselmanagements wird die Prüfung der Paritätsbits nicht empfohlen. Übertragungsfehler sind über das übergeordnete PKCS#1-Format erkennbar. Durch das Auslassen der Prüfung der Paritätsbits wird die Interoperabilität in den Fällen verbessert, in denen keine Übertragungsfehler vorlagen, der Sender jedoch entgegen der in [RFC1423] und in dieser Spezifikation getroffenen Festlegung keine korrekten Paritätsbits eingefügt hat.

Triple-DES-Schlüssel werden, auch in der verschlüsselten Form, stets in voller Länge, d.h. mit allen 128 Schlüsselbits einschließlich Paritätsbits übertragen.

Für den Triple-DES ist im CBC-Mode darüber hinaus ein 64-Bit langer Initialisierungsvektor (IV) erforderlich. Für jede zu verschlüsselnde Nachricht ist ein neuer Initialisierungsvektor zu generieren.

Beim Erzeugen verschlüsselter Nachrichten werden im „DEK-Info“-Feld zwei Argumente eingetragen. Das erste Argument lautet „DES3-CBC“ und referenziert den hier spezifizierten Algorithmus. In dem zweiten Argument wird der zugehörige Initialisierungsvektor eingetragen, hexadezimal kodiert in 16 Hexadezimalziffern.

Bei symmetrischem Schlüsselmanagement wird ausserdem der verschlüsselte Triple-DES-Schlüssel in hexadezimaler Kodierung als drittes Element im „Key-Info“-Feld eingetragen.

8.3 Schlüsselmanagement

Die nachfolgend beschriebenen Algorithmen werden dazu verwendet, Nachrichtenschlüssel und Hashwerte zu verschlüsseln bzw. zu signieren.

Für symmetrisches Schlüsselmanagement steht als Algorithmus der DES im ECB und im EDE-Modus zur Verfügung, für asymmetrisches Schlüsselmanagement der RSA.

Bei symmetrischem Schlüsselmanagement werden sowohl der Nachrichtenschlüssel (bei verschlüsselten Nachrichten) als auch der Hashwert der Nachricht (bei allen Nachrichten) verschlüsselt. Das Verschlüsseln des Hashwerts liefert einen Message Authentication Code (MAC), der wie eine digitale Signatur zur Überprüfung der Integrität und Authentizität einer Nachricht verwendet werden kann. Eine Sicherung der Verbindlichkeit von Nachrichten ist über diesen Mechanismus nicht möglich, da zumindest Sender und Empfänger über den gleichen symmetrischen Schlüssel verfügen[10] und damit beide gleichermaßen in der Lage sind, einen bestimmten MAC zu erzeugen. Eine eindeutige Zuordnung einer Nachricht zu einem bestimmten Sender ist damit auf diese Weise nicht möglich.

Bei asymmetrischem Schüsselmanagement wird der Nachrichtenschlüssel mit dem öffentlichen Schlüssel des dedizierten Empfängers verschlüsselt (bei verschlüsselten Nachrichten) und der Hashwert mit dem privaten Schlüssel des Senders signiert (bei allen Nachrichten). Bei verschlüsselten Nachrichten wird außerdem der signierte Hashwert zusätzlich mit dem öffentlichen Schlüssel des Empfängers verschlüsselt.

Bei symmetrischem Schlüsselmanagement wird für die Verschlüsselung des Nachrichtenschlüssels und des Hashwerts einer Nachricht derselbe Algorithmus und derselbe Key Encryption Key (KEK) verwendet (siehe 3.2.4.11 „Key-Info“).

Bei asymmetrischen Schlüsselmanagement werden für die Verschlüsselung des Nachrichtenschlüssels und für die Signatur des Hashwerts unterschiedliche KEK verwendet. Auch die Algorithmen können unterschiedlich sein, da die zugehörigen Informationen in zwei verschiedenen Kopffeldern („Key-Info“, „MIC-Info“) eingetragen sind.

Die gleichzeitige Verwendung eines symmetrischen Schlüsselmanagements für die Verschlüsselung von Nachrichtenschlüsseln und eines asymmetrischen Schlüsselmanagements zur Erzeugung einer digitalen Signatur ist innerhalb einer Nachricht aufgrund der Definition des „Key-Info“-Felds (siehe 3.2.4.11) nicht möglich.

---

10 Bei Gruppenschlüsseln können dies gegebenenfalls auch noch mehr Teilnehmer sein.

8.3.1 DES im ECB-Mode

- **Objekt-Identifier**:
  { iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) des-ecb(6) }
- **Bezeichnung in Kopfzeilen**: "DES-ECB"
- **ASN.1-Parameter im Typ „AlgorithmIdentifier“**: NULL
- **Verwendungszweck** (siehe Kapitel 8): encrypt, sign_data
- **Referenz und Kurzbeschreibung**:

  Der DES (Data Encryption Standard) ist standardisiert in [FIPS46] und in [X3.92]. Der Verschlüsselungsmodus ECB (Electronic Codebook Mode) beschreibt die Verwendung des Basisalgorithmis (hier: des DES) in direkter Form ohne weitere Rückkopplung und ist in [FIPS81] und [X3.106] standardisiert.

8.3.2 Triple-DES im ECB-Mode

- **Objekt-Identifier**:
  { iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) des-ede(17) }
- **Bezeichnung in Kopfzeilen**: "DES-EDE"
- **ASN.1-Parameter im Typ „AlgorithmIdentifier“**: NULL
- **Verwendungszweck** (siehe Kapitel 8): encrypt, sign_data
- **Referenz und Kurzbeschreibung**:

  Der DES im EDE-Modus ist in [X9.17] standardisiert. Für ihn wird im Gegensatz zum Standard DES ein Paar von 64-Bit-Schlüsseln benötigt, insgesamt also 128 Bit.

8.3.3 RSA

- **Objekt-Identifier**:
  { joint-iso-ccitt(2) ds(5) algorithm(8) encryptionAlgorithm(1) rsa(1) }
- **Bezeichnung in Kopfzeilen**: "RSA"
- **ASN.1-Parameter im Typ „AlgorithmIdentifier“**: KeySize
  Enthält die Länge des RSA-Modulus in Bit, kodiert als ASN.1 INTEGER.
- **Verwendungszweck** (siehe Kapitel 8): encrypt, sign_data
- **Referenz und Kurzbeschreibung**:

  Der RSA-Algorithmus, die Formatierung von RSA-Schlüsseln und die Aufbereitung von Daten vor Bearbeitung mit RSA sind in [PKCS1] spezifiziert.

  Die RSA-Moduluslänge ist bei Standard-PEM-Nachrichten („MIC-ONLY“, „MIC-CLEAR“, „ENCRYPTED“) auf Werte zwischen 508 und 1024 Bit beschränkt. Im Rahmen dieser Spezifikation wird dieser Bereich auf Werte zwischen 508 und 2048 Bit erweitert.

  Für Standard-PEM-Nachrichten ist als öffentlicher Exponent *e* entweder der Wert „3“ oder der Wert „65537“ (4. Fermatzahl“) zu verwenden. Abweichend von [RFC1423] wird im Rahmen dieser Spezifikation aus Sicherheitsgründen für alle Nachrichten, auch für Standard-PEM-Nachrichten, die Verwendung von „65537“ als öffentlichem Exponenten empfohlen. Für Nachrichten, die nicht PEM-kompatibel sein müssen, wird die Beschränkung der Wahl des öffentlichen Exponenten aufgehoben.

Daten müssen vor der Verarbeitung mit RSA nach dem in [PKCS1] spezifizierten Verfahren durch Padding zu Datenblöcken aufbereitet werden. Für die Verschlüsselung von Nachrichtenschlüsseln ist dabei PKCS#1-Blocktyp „02“, für die Signatur von Hashwerten PKCS#1-Blocktyp „01“ zu verwenden.

Bei der Verschlüsselung eines Nachrichtenschlüssels wird das Ergebnis der RSA-Verschlüsselung als zweiter Parameter in dem zugehörigen „Key-Info“-Feld (siehe 3.2.4.11) eingetragen, kodiert nach dem in 3.2.3.4 spezifizierten Verfahren.

Bei Signatur eines Hashwerts wird das Ergebnis der RSA-Transformation als dritter Parameter in dem entsprechenden „MIC-Info“-Feld (siehe 3.2.4.7) eingetragen, kodiert nach dem in 3.2.3.4 spezifizierten Verfahren.

8.3.4 RSA Verschlüsselung

- **Objekt-Identifier**:
  { iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) pkcs-1(1) rsaEncryption(1) }
- **Bezeichnung in Kopfzeilen**: "RSA"
- **ASN.1-Parameter im Typ „AlgorithmIdentifier“**: NULL
- **Verwendungszweck** (siehe Kapitel 8): encrypt
- **Referenz und Kurzbeschreibung**

  Die Beschreibung zu diesem Verfahren ist identisch mit der Beschreibung in Kapitel 8.3.3 „RSA“, eingeschränkt auf die Anwendung von RSA zur Verschlüsselung von Nachrichtenschlüsseln.

  Achtung ! In Standard-PEM-Nachrichten kann rsaEncryption auch zum Signieren von Hashwerten genutzt werden, was für andere Nachrichten im Rahmen dieser Spezifikation nicht vorgesehen ist.

8.3.5 RSA Signatur

- **Objekt-Identifier**:
  { iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) rsaSignature(11) }
- **Bezeichnung in Kopfzeilen**: "RSA"
- **ASN.1-Parameter im Typ „AlgorithmIdentifier“**: NULL
- **Verwendungszweck** (siehe Kapitel 8): sign_data
- **Referenz und Kurzbeschreibung**:

  Die Beschreibung zu diesem Verfahren ist identisch mit der Beschreibung in Kapitel 8.3.3 „RSA“, eingeschränkt auf die Anwendung von RSA zur Signatur von Hashwerten.

8.4 Hashen und Signieren von Zertifikaten und Sperrlisten

8.4.1 MD2 mit RSA

- **Objekt-Identifier**:
  { iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) pkcs-1(1) md2WithRSAEncryption(2) }
- **Bezeichnung in Kopfzeilen**: keine
- **ASN.1-Parameter im Typ „AlgorithmIdentifier“**: NULL
- **Verwendungszweck** (siehe Kapitel 8): sign_cert
- **Referenz und Kurzbeschreibung**:

  Die mit dem angegebenen Objekt-Identifier bezeichnete Kombination aus MD2 und RSA ist in [PKCS1] spezifiziert. Das RSA-Verfahren selbst ist ebenfalls in [PKCS1] spezifiziert, der MD2 ist in [RFC1319] spezifiziert.

8.4.2 MD5 mit RSA

- **Objekt-Identifier**:
  { iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) pkcs-1(1) md5WithRSAEncryption(4) }
- **Bezeichnung in Kopfzeilen**: keine
- **ASN.1-Parameter im Typ „AlgorithmIdentifier“**: NULL
- **Verwendungszweck** (siehe Kapitel 8): sign_cert
- **Referenz und Kurzbeschreibung**:

  Die mit dem angegebenen Objekt-Identifier bezeichnete Kombination aus MD5 und RSA ist in [PKCS1] spezifiziert. Das RSA-Verfahren selbst ist ebenfalls in [PKCS1] spezifiziert, der MD5 ist in [RFC1321] spezifiziert.

8.4.3 SHA-1 mit RSA

- **Objekt-Identifier**:
  { iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) sha1WithRSASignature(29) }
- **Bezeichnung in Kopfzeilen**: keine
- **ASN.1-Parameter im Typ „AlgorithmIdentifier“**: NULL
- **Verwendungszweck** (siehe Kapitel 8): sign_cert
- **Referenz und Kurzbeschreibung**:

  Die mit dem angegebenen Objekt-Identifier bezeichnete Kombination aus SHA-1 und RSA ist analog zu den oben beschriebenen Kombinationen aus MD2 mit RSA (siehe 8.4.1) bzw. MD5 mit RSA (siehe 8.4.2) zu realisieren. Die Kombination von MD2 mit RSA und MD5 mit RSA ist in [PKCS1] spezifiziert. Das RSA-Verfahren selbst ist ebenfalls in [PKCS1] spezifiziert, der SHA-1 ist in [FIPS180-1] spezifiziert.

9 Literaturverzeichnis

[FIPS46] Federal Information Processing Standards (FIPS PUB) 46-1: *Data Encryption Standard*, Januar 1988.

[FIPS81] Federal Information Processing Standards (FIPS PUB) 81: *DES Modes of Operation*, Dezember 1980.

[FIPS180-1] Federal Information Processing Standards (FIPS PUB) 180-1: *Secure Hash Standard*, Januar 1988.

[Hues95] Hueske, Thomas: *MailTrusT-Infrastrukturbeschreibung*, TeleTrusT, 1995.

[ISO10116] International Organization for Standardization: *Modes of Operation for an n-bit block cipher* algorithm. Standard ISO/IEC 10116, 1991.

[PKCS1] RSA Laboratories: *PKCS #1: RSA Encryption Standard*, Redwood City, November 1993.

[PKCS11] RSA Laboratories: *PKCS #11: Cryptographic Token Interface Standard*, Redwood City, April 1995.

[PKCS11a] RSA Laboratories: *PKCS #11 Errata*, August 1996.

[PKIX1] PKIX Working Group: *Internet Public Key Infrastructure, Part I: X.509 Certificate and CRL Profile*, Internet-Draft <draft-ietf-pkix-ipki-part1-02.txt>, Juni 1996.

[RFC1319] Kaliski, Burt: *The MD2 Message-Digest Algorithm*, Internet Engineering Task Force, RFC1319, April 1992.

[RFC1321] Rivest, Ron: *The MD5 Message-Digest Algorithm*, Internet Engineering Task Force, RFC1321, April 1992.

[RFC1421] Linn, J.: *Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures*, Internet Engineering Task Force, RFC1421, Februar 1993.

[RFC1422] Kent, S.: *Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management*, Internet Engineering Task Force, RFC1422, Februar 1993.

[RFC1423] Balenson, D.: *Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers*, Internet Engineering Task Force, RFC1423, Februar 1993.

[RFC1424] Kaliski, B.: *Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services*, Internet Engineering Task Force, RFC1424, Februar 1993.

[RFC821] Postel, Jonathan B.: *Simple Mail Transfer Protocol*, Internet Engineering Task Force, RFC821, August 1982.

[RFC822] Crocker, David H.: *Standard for the Format of ARPA Internet Text Messages*, Internet Engineering Task Force, RFC822, August 1982.

[X.208] International Telecommunication Union: *Specification of Abstract Syntax Notation One (ASN.1)*, ITU-T Recommendation X.208, 1993.

[X.209] International Telecommunication Union: *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*, ITU-T Recommendation X.209, 1993.

[X.509] Comitee Consultatif International Telegraphique et Telecommunication (CCITT): *The Directory - Authentication Framework*, Recommendation X.509, November 1988.

[X3.106] American National Standards Institute (ANSI): *Data Encryption Algorithm - Modes of Operation*, American National Standard for Information Systems X3.106, Mai 1983.

[X3.92] American National Standards Institute (ANSI): *Data Encryption Algorithm*, American National Standard X3.92, Dezember 1980.

[X9.17] American National Standards Institute (ANSI): *Financial Institution Key Management (Wholesale)*, American Bankers Association, American National Standard X9.17, April 1985.