

Hinweise zur Security Schnittstelle für das Gesundheits- und Sozialwesen

**Hinweise der Schnittstellen für die Übermittlung
von verschlüsselten und signierten Nachrichten**

Stand der Spezifikation: 10.10.2013

Version: 2.2.0

Herausgeber: GKV-Spitzenverband

Redaktion: Informationstechnische Servicestelle der
Gesetzlichen Krankenversicherungen GmbH
63150 Heusenstamm, Seligenstädter Grund 11
Telefon 06104/60050-0 - Telefax 06104/60050-300
E-Mail: info@itsg.de

Anmerkung: Die Technischen Spezifikationen sind Bestandteil der
technischen Anlagen der Beiträge und Richtlinien mit
den Leistungserbringern und den Arbeitgebern.

Änderungshistorie

Diese Darstellung beschreibt die Änderungen zur jeweiligen Vorgängerversion und soll eine kurze Information über die geänderten Teile des Dokumentes geben.

Version	2.2.0	gültig ab:		Datum:	10.10.2013
Seite	Punkt	Art der Änderung	Kurzbeschreibung der Änderung		
1		gelöscht	Hinweis auf alte Überschrift wurde entfernt		
3		gelöscht	Tabelle mit alter Änderungshistorie bis Version 1.5 gelöscht		
8	1	neu	AES Migrations-Strategie hinzugefügt		
51	7.4.4.2	neu	In Profilierung AES-Verschlüsselungsalgorithmus und Hinweis auf Migrations-Strategie hinzugefügt		
57	8.3	neu	AES-Verschlüsselungsalgorithmus mit OID hinzugefügt		
57	8.3	gelöscht	alter DES-Verschlüsselungsalgorithmus des3-cbc gelöscht		
69	10.1	neu	RFC des AES-Verschlüsselungsalgorithmus in Literaturverzeichnis hinzugefügt		
69	10.1	geändert	Version der Security Schnittstelle in Literaturverzeichnis aktualisiert		

Version	2.1.2	gültig ab:	01.04.2012	Datum:	01.06.2012
Seite	Punkt	Art der Änderung	Kurzbeschreibung der Änderung		
1		geändert	Die Überschrift wurde angepasst von „Hinweise zur Security Schnittstelle für das Gesundheitswesen“ in „Hinweise zur Security Schnittstelle für das Gesundheits- und Sozialwesen“)		
29	5.5.2	gelöscht	Die E-Mail-Adresse „ itsg-crq@atosorigin.com “ wurde bei dem Punkt „E-Mail an einen automatisierten Mail-Empfänger“ gelöscht		

Version	2.1.1	gültig ab:	01.04.2012	Datum:	10.04.2012
Seite	Punkt	Art der Änderung	Kurzbeschreibung der Änderung		
alle	alle	geändert	Anpassung der Referenzen		

Version	2.1.0	gültig ab:	01.04.2012	Datum:	19.01.2012
Seite	Punkt	Art der Änderung	Kurzbeschreibung der Änderung		
1		geändert	Herausgeber auf GKV-Spitzenverband geändert		
8	1	neu	Hinweis auf Migrationsschritte für SHA-256		
8	1	geändert	Nummern der Kapitelübersicht aktualisiert		
23, 32	5.2, 6.3.2	geändert	Neuer Hash-Algorithmus SHA-256 hinzugefügt		

**Hinweise
zur Security Schnittstelle
für das Gesundheits- und Sozialwesen**



Version	2.1.0	gültig ab:	01.04.2012	Datum:	19.01.2012
Seite	Punkt	Art der Änderung	Kurzbeschreibung der Änderung		
	Alle	geändert	Referenzen aktualisiert		

Version	2.0.0	gültig ab:	01.09.2011	Datum:	21.06.2011
Seite	Punkt	Art der Änderung	Kurzbeschreibung der Änderung		
	Alle	geändert	Redaktionelle Änderungen		
	3.2, 3.3.3 7.2.2, 8.1, 8.2	neu	Neuer Hash-Algorithmus SHA-256 hinzugefügt		
	9.1, 9.3, 9.4	gelöscht	Alle Punkte zu PEM gelöscht		
	3.3.6	neu	Zertifikatshinweis für Zahlstelle hinzugefügt		
1		neu	Der Punkt ‚Anmerkung‘ wurde hinzugefügt.		

Version	1.7.2	gültig ab:	28.07.2008	Datum:	
Seite	Punkt	Art der Änderung	Kurzbeschreibung der Änderung		
1	Alle	geändert	Redaktionelle Änderungen		
12	3.2	geändert	Ergänzungen zum Feld „algorithm“		
	3.3.2	geändert	Profilierung wurde um führende Nullen ergänzt.		
	4.2	geändert	„ .. signierte und verschlüsselte Nachricht ..“ in richtige Reihenfolge gebracht		
	5.4	neu	Erklärung einer PKCS#10 Zertifizierungsanfrage gemäß Kapitel 5		
	6	geändert	Eigenes Kapitel für PKCS#7 Zertifizierungsantwort (vorher Kapitel 7.2)		
	6.4	neu	Erklärung einer PKCS#7 Zertifizierungsantwort gemäß Kapitel 6		
	7	geändert	Eigenes Kapitel für PKCS#7 signierte und verschlüsselte Nachricht (vorher Kapitel 7.3)		
	7.3	neu	Erklärung einer signierten Nachricht gemäß Kapitel 7.2		
	7.5	neu	Erklärung einer PKCS#7-verschlüsselten Nachricht gemäß Kapitel 7.4		
	8.3	geändert	Aktuellen Triple-DES-Algorithmus des-ede3-cbc ergänzt		
	8.4	geändert	Referenz auf richtige Tabelle 5 gesetzt		
	9	geändert	Das Kapitel LDAP wurde komplett überarbeitet, dabei sind wesentliche Punkte aus dem LDAP-Workshop übernommen worden.		

Inhaltsverzeichnis

1	ZUSAMMENFASSUNG	8
1.1	Definition von Zertifikatsanfragen nach PKCS#10	8
1.1.1	Zertifikate X.509v3	8
1.2	Abgleich mit der Security Schnittstelle.....	9
2	ALLGEMEINE GRUNDLAGEN	10
2.1	Verwendung von Schlüsselworten.....	10
2.2	Definitionen.....	10
3	ZERTIFIKATE X.509V3	12
3.1	Signatur	12
3.2	Identifizierung des verwendeten Signaturalgorithmus	12
3.3	Zu signierendes Zertifikat	13
3.3.1	Versionsnummer	13
3.3.2	Seriennummer	14
3.3.3	Signaturalgorithmus	14
3.3.4	Name der Zertifizierungsstelle	15
3.3.5	Gültigkeitsdauer	17
3.3.6	Namen von Zertifikatsinhabern	18
3.3.7	Basic Constraints.....	19
3.3.8	Öffentlicher Schlüssel des Zertifikatsinhabers	19
3.3.9	Nicht verwendete optionale Datenfelder	20
4	NACHRICHTENAUSTAUSCH.....	21
4.1	PKI-Nachrichten.....	21
4.2	signierte und verschlüsselte Nachrichten.....	21
5	PKCS#10-ZERTIFIZIERUNGSANFRAGE.....	22
5.1	Überblick	22
5.2	Aufbau des PKCS#10-Datentyps.....	22
5.3	Aufbau der Teildatenstruktur CertificationRequestInfo.....	23
5.3.1	Versionsnummer	23
5.3.2	Namen von Zertifikatsinhabern	23
5.3.3	Öffentlicher Schlüssel des Zertifikatsinhabers	24
5.3.4	Profilierung: Die Angaben in Abschnitt 3.3.7 „Basic Constraints.....	24
5.3.5	Erweiterungen	24

5.4	Erklärung einer PKCS#10 Zertifizierungsanfrage gemäß Kapitel 5	25
5.4.1	Teil 1, Version und Subject Name	26
5.4.2	Teil 2, PublicKeyInfo	27
5.4.3	Teil 3, CertificationRequest.Signature	28
5.5	Transport der PKCS#10 Zertifizierungsanfrage	29
5.5.1	Transportformat	29
5.5.2	Transportwege	29
6	PKCS#7-ZERTIFIZIERUNGSANTWORT	30
6.1	Überblick PKCS#7	30
6.2	Aufbau der PKCS#7-Zertifizierungsantwort	31
6.3	Aufbau der Teildatenstruktur Content vom ContentType „SignedData“	32
6.3.1	Datenfeld „version“	32
6.3.2	Datenfeld „digestAlgorithms“	32
6.3.3	Datenfeld „encapContentInfo“	32
6.3.4	Datenfeld „certificates“	33
6.3.5	Datenfeld „crls“	33
6.3.6	Datenfeld „signerInfos“	33
6.4	Erklärung einer PKCS#7 Zertifizierungsantwort gemäß Kapitel 6	34
6.4.1	Teil 1, Version und Subject Name	34
6.4.2	Teil 2a, PublicKeyInfo	35
6.4.3	Teil 2b, PublicKeyInfo	36
6.4.4	Teil 2c, PublicKeyInfo	37
6.4.5	Teil 3, IssuerKeyInfo	38
6.4.6	Teil 4, PCAKeyInfo	39
6.5	Transport der PKCS#7-Zertifizierungsantwort	40
6.5.1	Transportformat	40
6.5.2	Transportwege	40
7	PKCS#7-SIGNIERTE UND VERSCHLÜSSELTE NACHRICHT	41
7.1	Aufbau der PKCS#7-signierten und verschlüsselten Nachricht	41
7.2	Aufbau der Teildatenstruktur Content vom ContentType „SignedData“	42
7.2.1	Datenfeld „version“	42
7.2.2	Datenfeld „digestAlgorithms“	42
7.2.3	Datenfeld „encapContentInfo“	42
7.2.4	Datenfeld „certificates“	42
7.2.5	Datenfeld „crls“	43
7.2.6	Datenfeld „signerInfos“	43
7.3	Erklärung einer signierten Nachricht gemäß Kapitel 7.2	45
7.3.1	Teil 1, die signierende Datei	45
7.3.2	Teil 2, signerInfos	46
7.3.3	Teil 2a, signedAttrs	47
7.4	Aufbau der Teildatenstruktur Content vom ContentType „EnvelopedData“	48
7.4.1	Datenfeld „version“	48

7.4.2	Datenfeld „originatorInfo“	48
7.4.3	Datenfeld „RecipientInfos“	48
7.4.4	Datenfeld „encryptedContentInfo“	50
7.4.5	Datenfeld „unprotectedAttrs“ (entfällt!)	51
7.5	Erklärung einer PKCS#7-verschlüsselten Nachricht gemäß Kapitel 7.4.....	52
7.5.1	Teil 1, envelopedData	52
7.5.2	Teil 2, OriginatorInfo und RecipientInfo.....	53
7.5.3	Teil 3, EncryptedContentInfo	54
7.6	Transport der der PKCS#7-verschlüsselten Nachricht.....	55
7.6.1	Transportformat.....	55
7.6.2	Transportwege.....	55
8	KRYPTO-ALGORITHMEN.....	56
8.1	Einweg-Hash-Funktionen (One-Way Hash Functions).....	56
8.2	Signaturalgorithmen (Signature Algorithms)	57
8.3	Verschlüsselungsalgorithmen für Daten (Content Encryption Algorithms)	57
8.4	Verschlüsselungsalgorithmen für den Nachrichtenschlüssel (Key Encryption Algorithms).....	58
8.5	Algorithmen zur Nutzung des öffentlichen Schlüssels (Subject Public Key Algorithms).....	58
9	LDAP	60
9.1	Aufbau des LDAP Servers	60
9.2	Aufbau der Wurzel.....	60
9.3	Aufbau des Trust Center Zweigs (geordnet nach Datenaustauschverfahren)	61
9.3.1	Aufbau der PCA- und CA-Ebene	61
9.4	Aufbau des Zertifikatzweigs (geordnet nach Herausgeber).....	61
9.4.1	Aufbau der Issuer-Ebene	62
9.4.2	Aufbau der Zertifikats-Ebene	62
9.5	Aufbau des Alias-Zweigs.....	62
9.5.1	Aufbau der PCA-, CA-, AG-, LE- und Annahme-Ebene.....	63
9.5.2	Aufbau der Verweis-Ebene	63
9.6	Aufbau des Update-Zweig.....	63
9.6.1	Aufbau der Datums-Ebene	64
9.7	Zugriffsrechte und -rollen	64
9.7.1	Anonyme Zugriffsrechte.....	64
9.7.2	Anmeldung mit Zugriffsrechten (Annahmestellen der GKV).....	65
9.8	Suchfunktionen	65
9.9	Zertifikatshandling	65
9.9.1	Sperrung eines Zertifikats.....	65

9.9.2	Abgelaufene Zertifikate.....	65
9.9.3	Abholung eines einzelnen Zertifikats vom LDAP-Server	66
9.9.4	Abholung einer Sperrliste vom LDAP-Server.....	66
9.10	Betrieb des LDAP Servers	67
9.11	Eindeutiger LDAP-Server zum Abruf aller Zertifikate	67
10	ANHANG	68
10.1	Literaturverzeichnis	68
10.2	Abkürzungsverzeichnis	69

1 Zusammenfassung

Das vorliegende Dokument enthält eine Feinspezifikation als Ergänzung zur „Security Schnittstelle für das Gesundheits- und Sozialwesen“ der Version 2.1.0 [SecSchnitt]. Es schreibt die technische Ausgestaltung der Kommunikation auf der Basis des ISIS-MTT-Standards V1.0.2 vom 19. Juli 2002 fest.

Die Feinspezifikation dient als technische Geschäftsgrundlage mit Herstellern und Dienstleistern im Datenaustausch für das Gesundheitswesen. Bezüglich der Migrations-Strategie von SHA1 auf SHA-256 Algorithmus und Triple-DES auf AES sind die Migrationsschritte in der Einleitung der Spezifikation „Security Schnittstelle für das Gesundheitswesen“ Version 2.1.0 zu beachten.

Die Dokumentation ist in folgende Kapitel gegliedert:

- Das Kapitel 1 enthält Definitionen von Begriffen, deren exakte Bedeutung für das Verständnis wesentlich ist.
- Im Kapitel 2 werden wichtige Grundlagen für dieses Dokument festgelegt.
- Im Kapitel 3 wird ein Profil für die Zertifikate einer PKI „Datenaustausch im Gesundheitswesen“ nach „Security Schnittstelle für das Gesundheitswesen“ definiert.
- Im Kapitel 4 „Nachrichtenaustausch“ wird ein Überblick über die Bildung signierter und verschlüsselter Nachrichten nach PKCS#10 und PKCS#7 gegeben.
- Im Kapitel 5 werden Zertifikatsanfragen nach PKCS#10 beschrieben.
- Im Kapitel 6 werden Zertifikatsantworten beschrieben.
- Im Kapitel 7 werden signierte und verschlüsselte Nachrichten (E-Mail) nach PKCS#7 beschrieben.
- Das Kapitel 8 stellt die verwendeten Signatur- und Verschlüsselungsverfahren dar.
- Im Kapitel 9 wird der LDAP-Dienst beschrieben, der vom Trust Center alle Zertifikate bereitstellt.
- Der Anhang im Kapitel 10 verweist auf die genutzten Standarddokumente.

1.1 Definition von Zertifikatsanfragen nach PKCS#10

Anhand des Standards PKCS#10 wird ein Profil erarbeitet, das [SecSchnitt] entspricht. PKCS#10 (Certification Request Syntax Standard) beschreibt eine Syntax für Zertifikatsanfragen; vgl. [PKCS#10]. Das technische Profil für Zertifikatsanfragen nach PKCS#10 ist im Kapitel 5 beschrieben.

1.1.1 Zertifikate X.509v3

X.509 ist eine Empfehlung der ITU-T Recommendation. Sie spezifiziert die Authentifizierungsdienstleistung für X.500-Verzeichnisse und die weit verbreitete X.509-Zertifikatsstruktur. Seit Version 3 (1993) sind Sicherheitsprobleme behoben, die in Version 1 und 2 noch bestanden. X.509 spezifiziert keine bestimmten kryptographischen Algorithmen, doch wird im Anhang der RSA-Algorithmus beschrieben und damit propagiert.

Anhand des X.509-Standards wird ein Profil erarbeitet, das [SecSchnitt] entspricht. Das technische Profil für X.509v3-Zertifikate ist im Kapitel 3 beschrieben.

1.2 Abgleich mit der Security Schnittstelle

In diesem Abschnitt werden Einschränkungen aufgeführt, da die Feinspezifikation nicht allen Vorgaben der Security Schnittstelle folgt.

Dieses gilt insbesondere für die Vorgaben hinsichtlich des Signaturgesetzes in [SecSchnitt] Abschnitt 1.1 „Einleitung“, die Forderungen der o. a. Security Schnittstelle nach qualifizierten Zertifikaten mit Anbieterakkreditierung wird derzeit nicht gestellt. Hinsichtlich der digitalen Umschläge für PKCS#10-Zertifizierungsanfragen und für PKCS#10-Zertifizierungsanfragen wird keine MIME-Einbettung im Sinne eines digitalen Umschlags verwendet.

PKCS#7-Objekte:

- Für PKCS#7- Zertifizierungsantworten wird keine MIME-Einbettung im Sinne eines digitalen Umschlags verwendet.
- Für PKCS#7- verschlüsselte Nachrichten wird keine MIME-Einbettung im Sinne eines digitalen Umschlags verwendet.

2 Allgemeine Grundlagen

2.1 Verwendung von Schlüsselworten

Für die genaue Unterscheidung zwischen der Verbindlichkeit und Aussagekraft von Inhalten und Vorgaben werden entsprechende Schlüsselworte in deutscher Sprache verwendet. Zu den folgenden Schlüsselworten ist jeweils die Mehrzahl eingeschlossen:

- MUSS, IST, WIRD oder FORDERT bedeutet, dass es sich um eine absolutgültige Festlegung bzw. Anforderung handelt. Die verbindlichen Vorgaben sind mit einer Profilierung in grauen Anmerkungsfelder zu den einzelnen Punkten aufgeführt
- DARF NICHT, IST NICHT, WIRD NICHT, ENTFÄLLT oder VERWENDET KEINE bezeichnet den absolut gültigen Ausschluss einer Festlegung bzw. Anforderung.
- SOLL oder VORSCHLAG beschreibt eine Empfehlung. Abweichungen zu diesen Festlegungen sind in begründeten Fällen möglich.
- SOLL NICHT kennzeichnet die Empfehlung, eine Eigenschaft auszuschließen. Abweichungen sind in begründeten Fällen möglich
- KANN oder OPTIONAL bedeutet, dass die Eigenschaften fakultativ oder optional sind und damit keinen allgemeingültigen Empfehlungscharakter besitzen.

2.2 Definitionen

Basic Encoding Rules (BER)

Transfersyntax zur Kodierung ASN.1-Strukturen in binäre Darstellung, definiert in X.209 (X.690).

Distinguished Encoding Rules (DER)

Untermenge der BER zur eindeutigen Kodierung ASN.1-Strukturen in binärer Darstellung, definiert in X.690.

MIME (Multipurpose Internet Mail Extensions)

MIME ist ein Kodierungsverfahren für die Einbindung von binären Daten in Internet-Mails. Zusätzlich unterstützt MIME Multipart-Mails, um in einer Mail verschiedene Datentypen oder binäre Anhänge und Mails im HTML-Format zu ermöglichen.

Objektbezeichner (Object Identifier, OID)

Ein Objektbezeichner ist ein Datentyp, der ein eindeutig festgelegtes Objekt bezeichnet. Er ist eine Folge von nicht negativen, ganzzahligen Werten, die einen bestimmten Pfad durch eine Baumstruktur markieren. Der Baum besteht aus einer Wurzel, die mit einer Reihe von gekennzeichneten Knoten über Äste verbunden ist. Jeder Knoten kann wiederum weitere, eigene Unterknoten haben, die man auch Unterbäume nennt. Objektbezeichner stellen die Möglichkeit dar, ein Objekt unabhängig von dessen Semantik zu bestimmen. Für das Beschreiben eines OID kann man mehrere Formate verwenden. Das genaueste Textformat ist die Auflistung der beim Durchlaufen des Baumes - von der Wurzel über die Knoten in Richtung des in Frage kommenden Objekts – gefundenen ganzen Zahlen. Die ganzen Zahlen werden mit einem Punkt voneinander getrennt.

Beispiel: OID 1.2.840.113549.1.10.1.1

ASN.1-Struktur für PKCS#10

Profil

Durch die Bildung eines Profils werden Standards, die für eine Vielzahl unterschiedlicher Anwendungen entworfen wurden, auf einen konkreten Anwendungsbereich abgebildet. Standards bieten oft eine Vielzahl von Optionen, die sich nicht selten widersprechen. Für jeden Anwendungsbereich muss deshalb eine sorgfältige Auswahl der Optionen getroffen werden. Außerdem muss allen Objekten des Standards durch das Profil eine eindeutige Semantik zugeordnet werden, sofern Mehrdeutigkeiten bestehen. Es muss auch festgelegt werden, ob die Unterstützung durch konforme Komponenten zwingend ist. Nur so kann der Standard interoperabel gemacht werden.

Request For Comment (engl.) - Bitte um Kommentar

Ein RFC ist ein Papier (im übertragenen Sinne), das von der Internet Engineering Task Force (IETF) mit der Bitte um Kommentar veröffentlicht wird. Ein solches Papier stellt Vorschläge für Standards vor, die im Internet verbindlich werden sollen. Je nach Inhalt der Reaktionen (Kommentare) werden die Vorschläge abgewandelt oder verbindlich erklärt.

RSA (Rivest-Shamir-Adleman)

1978 vorgestellter Algorithmus der Klasse der asymmetrischen Chiffren, der u. a. zur Erstellung digitaler Signaturen eingesetzt wird.

3 Zertifikate X.509v3

Referenz: [SecSchnitt], Abschnitte 2.1.7 und 3.2.8
[IMTTP1], Kapitel 2, Tabelle 1

Da die oben angeführte Security Schnittstelle ISIS-MTT als Spezifikationsgrundlage voraussetzt, werden im Folgenden alle ASN.1-Syntaxangaben hieraus verwendet.

Ein X.509v3-Zertifikat besteht aus einer Folge von Feldern und hat die folgende Struktur:

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

Das Feld `tbsCertificate` enthält das eigentliche X.509-Zertifikat, das signiert wird (`tbs` steht für *to be signed*). Das Feld `signatureAlgorithm` dient der Identifizierung des verwendeten Signaturalgorithmus und das Feld `signature` enthält die Signatur des Zertifikats. Auf diese Felder wird im Folgenden wegen des ansteigenden Informationsgehalts in umgekehrter Reihenfolge eingegangen.

In X.509v3 können Zertifikatserweiterungen (Extensions) verwendet werden, um einem Zertifikat Informationen verschiedenster Art beizufügen. Erweiterungen können eine Vielzahl zusätzlicher Informationen enthalten, z. B. Angaben über den Teilnehmer, den Schlüssel oder Beschränkungen des Zertifizierungspfades.

Profilierung:

Die vorliegende Spezifikation verwendet keine Erweiterungen für Teilnehmerzertifikate.

3.1 Signatur

Referenz: [IMTTP1], Kapitel 2, Tabelle 1

Das Feld `signature` enthält die Signatur, die durch Anwendung des Algorithmus und des privaten Schlüssels der CA auf die zu signierenden Daten gebildet wird¹.

3.2 Identifizierung des verwendeten Signaturalgorithmus

Referenz: [SecSchnitt], Abschnitte 2.1.2, 2.1.3 und 3.2.3
[IMTTP1], Kapitel 2, Tabellen 1 und 4

Der Signaturalgorithmus `signatureAlgorithm` wird durch den Objektbezeichner (`Object Identifier`, `OID`) für den verwendeten Algorithmus und die verwendeten Parameterwerte identifiziert.

Es wird die folgende Struktur gebildet:

```
AlgorithmIdentifier ::= SEQUENCE {  
    algorithm OBJECT IDENTIFIER,  
    parameters ANY DEFINED BY algorithm OPTIONAL }
```

¹ Der Wert, der sich aus der DER-Kodierung des Inhaltes des Feldes „`tbsCertificate`“ ergibt, wird durch eine Hash-Funktion transformiert. Auf das Ergebnis werden der Verschlüsselungsalgorithmus und der private Schlüssel der CA angewandt. Das als ASN.1-Bitstring kodierte Ergebnis ist die Signatur. Die Einzelheiten der Signaturbildung werden in Kapitel 8 dargestellt.

Im Feld `algorithm` wird ein Algorithmus angegeben, der eine Kombination von Hash-Funktion und Verschlüsselungsalgorithmus ist. Das Feld `parameters` ist optional, da es nur bei Algorithmen von Bedeutung ist, für die Parameter angegeben werden müssen. Auch wenn für die Anwendung des Algorithmus ein Parameter angegeben werden muss, darf das Feld `parameters` nicht zur Übergabe dieses Parameters verwendet werden. Eine dem ISIS-MTT-Profil entsprechende Anforderung der vorliegenden Spezifikation ist, dass das Feld nur entweder ausgelassen werden (nur bei Zertifikaten!) oder den (ASN.1-)Wert NULL enthalten darf.

In Kapitel 8 werden die in diesem vorgesehenen `AlgorithmIdentifier` zusammen mit dem Zweck beschrieben, für den der jeweilige Algorithmus verwendet werden darf (Signieren, Verschlüsseln).

Profilierung:

Die Inhalte der Datenfelder `signatureAlgorithm` (aus der Datenstruktur `Certificate`) und `signature` (aus der Datenstruktur `tbsCertificate`) müssen identisch sein. Die Security Schnittstelle legt fest, dass SHA-1 bzw. SHA-256 mit RSA einzusetzen ist; vgl. Abschnitt 8.2 Signaturalgorithmen. Das Feld `parameters` ist mit dem Wert NULL zu belegen.

3.3 Zu signierendes Zertifikat

Referenz: [SecSchnitt], Abschnitt 3.2
[IMTTP1], Kapitel 2, Tabellen 1 und 4

Das Feld `tbsCertificate` besteht aus einer Folge von Teilfeldern, die die zu signierenden Daten enthalten:

```
TBSCertificate ::= SEQUENCE {
    version [0] EXPLICIT Version DEFAULT v1,
    serialNumber CertificateSerialNumber,
    signature AlgorithmIdentifier,
    issuer Name,
    validity Validity,
    subject Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo
    issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL
        (if present, version MUST be v2 or v3)
    subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL
        (if present, version MUST be v2 or v3)
    extensions [3] EXPLICIT Extensions OPTIONAL
        (if present, version MUST be v3)
```

Im Folgenden werden diese Felder beschrieben, wobei zu jedem Feld folgende Informationen angegeben und erläutert werden:

- Semantik des Feldes,
- die ASN.1-Struktur,
- die Bedeutung der einzelnen Teilfelder,
- Auswahl und Belegung der Teilfelder,
- Spezielle Interoperabilitätsanforderungen.

3.3.1 Versionsnummer

Referenz: [SecSchnitt], Abschnitt 3.2.1
[IMTTP1], Kapitel 2, Tabelle 2.#12

Die Versionsnummer identifiziert die Version des Zertifikates. Das Feld `version` könnte entsprechend den drei standardisierten Zertifikatsversionen nach ISS-MTT einen von drei Werten enthalten:

```
Version ::= INTEGER { v1(0), v2(1), v3(2) }
```

und kennzeichnet die Zertifikatsformate X.509v1, X.509v2 und X.509v3, d. h. die Integerwerte „0“, „1“ oder „2“ sind entsprechend angegeben.

Profilierung:

Es sind ausschließlich X.509v3-Zertifikate zu verwenden, der Integerwert für das Datenfeld `version` beträgt „2“.

3.3.2 Seriennummer

Referenz: [SecSchnitt], Abschnitt 3.2.2
[IMTTP1], Kapitel 2, Tabelle 2.#3

Die Seriennummer dient dazu, ein Zertifikat (als Element der Menge aller von einer CA erstellten Zertifikate) eindeutig zu identifizieren. Das Paar - bestehend aus Seriennummer (`serialNumber`) und Name der ausstellenden CA (`issuer`) - muss innerhalb einer PKI stets eindeutig sein, da es z. B. zur Identifizierung von Zertifikaten in Sperrlisten verwendet wird.

Dieses ist insbesondere wichtig zu berücksichtigen, wenn in zwei verschiedenen Zertifikaten (sprich: das eindeutige Paar aus Seriennummer und CA ist ungleich) der gleiche DN (`Distinguished Name`) vorhanden ist.

Beispiel²: Ein gleicher DN in zwei Zertifikaten kann entstehen, wenn für das gleiche Verfahren für die gleiche Institution ein und derselbe Mitarbeiter ein zweites Zertifikat beantragt und dieses erhält. Auch wenn der DN in beiden Zertifikaten gleich ist, sind es doch völlig unterschiedliche Zertifikate, da sie unterschiedliche Seriennummern haben. Dies ist selbst dann richtig, wenn die Zertifikate sogar auch noch denselben Gültigkeitszeitraum haben.

Die Seriennummer wird als natürliche Zahl dargestellt:

```
CertificateSerialNumber ::= INTEGER
```

Profilierung:

In Übereinstimmung mit dem ISIS-MTT-Profil müssen Seriennummern mit einer Länge von bis zu 20 Byte unterstützt werden, sie sind immer positiv. Für die Seriennummern steht der Zahlenraum von 1 ... 2¹⁵⁹ zur Verfügung, das verbleibende Bit (`Most Significant Bit`) ist zur Darstellung des positiven Vorzeichens zu verwenden. Führende Nullen (Ziffer) sind gestattet.

3.3.3 Signaturalgorithmus

Referenz: [SecSchnitt], Abschnitte 2.1.2, 2.1.3 und 3.2.3
[IMTTP1], Kapitel 2, Tabelle 2.#3

² Dieses Beispiel entstammt der bisherigen Praxis im ITSG-Trust Center. Das hier beschriebene Vorgehen ist nicht vereinbar mit den Regelungen des Signaturgesetzes zu den Angaben in einem qualifizierten Zertifikat: SigG §7 Absatz (1) besagt, dass der Name des Inhabers eines Zertifikats unverwechselbar sein muss, was ggf. durch einen Zusatz zum Namen erreicht wird.

Das Signatur-Datenfeld `signature` dient der Bezeichnung des Algorithmus, mit dem die Signatur gebildet wird³. Die Struktur ist mit der des Feldes `signatureAlgorithm` identisch (s. Abschnitt 3.2). Auch die Werte beider Felder müssen identisch sein; vgl. ISIS-MTT, Tabelle 4, Notes 4. Dies entspricht [RFC 2459].

```
AlgorithmIdentifier ::= SEQUENCE {  
    algorithm OBJECT IDENTIFIER,  
    parameters ANY DEFINED BY algorithm OPTIONAL }
```

Profilierung:

In Übereinstimmung mit dem ISIS-MTT-Profil müssen Seriennummern mit einer Länge von bis zu 20 Byte unterstützt werden, sie sind immer positiv. Für die Seriennummern steht der Zahlenraum von $1 \dots 2^{159}$ zur Verfügung, das verbleibende Bit (Most Significant Bit) ist zur Darstellung des positiven Vorzeichens zu verwenden. Führende Nullen (Ziffer) sind gestattet.

Profilierung:

In Übereinstimmung mit dem ISIS-MTT-Profil müssen Seriennummern mit einer Länge von bis zu 20 Byte unterstützt werden, sie sind immer positiv. Für die Seriennummern steht der Zahlenraum von $1 \dots 2^{159}$ zur Verfügung, das verbleibende Bit (Most Significant Bit) ist zur Darstellung des positiven Vorzeichens zu verwenden. Führende Nullen (Ziffer) sind gestattet.

Profilierung:

Die Inhalte der Datenfelder `signatureAlgorithm` (aus der Datenstruktur `Certificate`) und `signature` (aus der Datenstruktur „`tbsCertificate`“) müssen identisch sein. Die Security Schnittstelle legt fest, dass SHA-1 bzw. SHA-256 mit RSA einzusetzen ist; vgl. Abschnitt 8.2. Das Feld `parameters` ist mit dem Wert NULL zu belegen.

3.3.4 Name der Zertifizierungsstelle

Referenz: [SecSchnitt], Abschnitte 2.2 und 3.2.4
[IMTTP1], Kapitel 2, Tabellen 2.#5, 5, 6 und 15

Das Datenfeld `issuer` gibt den eindeutigen Namen des Zertifikaterzeugers (der CA) an. Er muss stets entsprechend der X.500-Syntax als `Distinguished Name` (DN) angegeben werden. Der Wert NULL ist nicht erlaubt.

`issuer` ist vom Datentyp „Name“. X.500-Distinguished-Names (kurz: DN) müssen folgender Syntax entsprechen:

```
Name ::= CHOICE { RDNSequence }  
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName  
RelativeDistinguishedName ::= SET OF AttributeTypeAndValue  
AttributeTypeAndValue ::= SEQUENCE {  
    typ AttributeType,  
    value AttributeValue }  
AttributeType ::= OBJECT IDENTIFIER  
AttributeValue ::= ANY DEFINED BY AttributeType
```

Der Datentyp `Name` besteht aus einer Folge von `RelativeDistinguishedNames`. Die Syntax sieht vor, dass ein `RelativeDistinguishedName` aus mehreren Komponenten vom Typ `AttributeTypeAndValue` bestehen kann. Entgegen der Syntax darf jedoch von dieser Möglichkeit

³ Es darf nicht mit dem gleichnamigen Feld verwechselt werden, das die Signatur enthält (s. o).

gemäß [IMTTP1] kein Gebrauch gemacht werden. Die Menge der `AttributeTypeAndValue`-Komponenten eines `RelativeDistinguishedName` darf nur ein Element enthalten. [IMTTP1] verlangt weiterhin,

- dass der Inhalt des Datenfelds `issuer` identisch ist mit dem Datenfeld `subject` in dem Zertifikat der CA,
- das Datenfeld `issuer` muss mindestens die Attribute `CountryName` und `Organization` enthalten.

Der DN besteht je nach Verwendungszweck des Zertifikats aus verschiedenen Attributen. Im vorliegenden Profil lassen sich gemäß Abschnitt 2.2 der Security Schnittstelle zwei Typen von DNs unterscheiden:

- allgemeine Teilnehmerzertifikate und
- Zertifizierungsstellen (PCA oder CA).

Da für das Feld `issuer` nur Zertifikate für Zertifizierungsstellen relevant sind, werden hierzu die beiden folgenden Tabellen angeführt:

Aufbau des DNs für Zertifizierungsstellen (CA)

Pos.	Attribute		Erläuterung
1	CountryName (verpflichtend)	C	Zweistelliges Kürzel für die Länderkennung, wie "DE" für Deutschland.
2	OrganizationName (verpflichtend)	O	Name des Trust Centers als Zeichenkette. Beispiel: „ITSG TrustCenter fuer den Datenaustausch im Gesundheitswesen“

Aufbau des DNs für Zertifizierungsstellen (PCA)

Pos.	Attribute		Erläuterung
1	CountryName (verpflichtend)	C	Zweistelliges Kürzel für die Länderkennung, wie "DE" für Deutschland.
2	OrganizationName (verpflichtend, fest)	O	Name der PCA als feste Zeichenkette: „Datenaustausch im Gesundheits- und Sozialwesen“

In welcher Reihenfolge die Attribute im Namen enthalten sind, ist für dieses Profil festgelegt. Die Reihenfolge der durch die Spalte "Pos." angegebenen Position der Attribute muss eingehalten werden.

Die möglichen Werte für ein Attribut werden durch die Typangabe bestimmt. Die Werte für die verwendeten Attribute sind wie folgt:

Attribute	Typangabe	Tabelle 7 in ISIS-MTT V1.0.2 Part 1	Maximale Länge der Zeichenkette
CommonName	PrintableString	# 1	128
OrganizationName	PrintableString	# 6	64
OrganizationalUnitName	PrintableString	# 7	64

Attribute	Typangabe	Tabelle 7 in ISIS-MTT V1.0.2 Part 1	Maximale Länge der Zeichenkette
CountryName	PrintableString (Size2)	# 13	2

Der Typ `PrintableString` ist eine Untermenge der Zeichensatz-Auswahl, die mit `DirectoryString` zur Verfügung steht. `DirectoryString` ist als Auswahl zwischen den folgenden Zeichensätzen definiert:

```
DirectoryString ::= CHOICE {  
  printableString PrintableString (SIZE (1..maxSize)),  
  teletexString TeletexString (SIZE (1..maxSize)),  
  utf8String UTF8String (SIZE (1..maxSize)),  
  bmpString BMPString (SIZE (1..maxSize)),  
  universalString UniversalString (SIZE (1..maxSize)) }
```

Hinweis:

ISIS-MTT definiert einen speziellen Zeichensatz als Teilmenge des Zeichensatzes UTF8String⁴, der es ermöglicht, alle Zeichen des Zeichensatzes US-ASCII und ISO-8859-1 (Latin-1) darzustellen. Diese Teilmenge soll als ISIS-MTT-UTF8String bezeichnet werden. Zeichen dieser Teilmenge werden durch ein oder zwei Byte dargestellt. Die Darstellung jedes Zeichens des Zeichensatzes 7-Bit-US-ASCII bleibt identisch erhalten. Diese Zeichen können weiterhin durch ein Byte dargestellt werden. UTF8String wird **in Ergänzung zur „Security Schnittstelle für das Gesundheitswesen Version 2.0.1 nicht gefordert**, obwohl ISIS-MTT ("must") dieses für alle Zertifikate fordert, die **nach dem 31. Dez 2003** ausgestellt werden.

Der Wert für „maxSize“ ist nicht in der Datenstruktur „SIZE (1..maxSize)“ selbst festgelegt, sondern wie oben bereits ausgeführt in Tabelle 7 der [IMTTP1].

Profilierung:

Die oben in den Tabellen ausgeführten Angaben zum DN und zu den Längen der Attribute des DNs sind verbindlich.

3.3.5 Gültigkeitsdauer

Referenz: [SecSchnitt], Abschnitte 3.2.5 und 5.1.2
[IMTTP1], Kapitel 2, Tabelle 2.#6 und 3

Durch das Feld `validity` wird angegeben, für welchen Zeitraum das Zertifikat gültig ist. Das Teilfeld `notBefore` gibt den Anfang und das Teilfeld `notAfter` das Ende des Gültigkeitszeitraums an, wobei beide Zeitpunkte in den Gültigkeitszeitraum eingeschlossen sind.

Die Gültigkeitsdauer des Zertifikates wird durch folgende Datenstrukturen definiert:

```
Validity ::= SEQUENCE {  
  notBefore Time,  
  notAfter Time }  
Time ::= CHOICE {  
  utcTime UTCTime,  
  generalizedTime GeneralizedTime }
```

⁴ UTF steht für Universal Character Set (UCS) Transformation Format. UTF-8 ist eine 8-Bit-Kodierung variabler Länge (s. [RFC 2279 98]).

Profilierung:

Bei der Generierung von Zertifikaten ist gemäß Security Schnittstelle `GeneralizedTime` zu verwenden, d. h. die Verwendung von `UTCTime` ist verboten⁵.

Die Datums- und Zeitangaben müssen im Format `YYYYMMDDHHMMSSZ`⁶ erfolgen, das in folgender Tabelle beschrieben wird:

Datumsangaben		Zeitangaben	
Feld	Bedeutung	Feld	Bedeutung
YYYY	vollständige Jahreszahl	HH	Stunde: 00, 01, ..., 23
MM	Monat 01, 02, ..., 12	MM	Minute 00, 01, ..., 59
DD	Tag 01, 02, ..., 31	SS ⁷	Sekunde 00, 01, ..., 59

3.3.6 Namen von Zertifikatsinhabern

Referenz: [SecSchnitt], Abschnitt 3.2.6
[IMTTP1], Kapitel 2, Tabelle 2.#7 und 5

Der Name im Feld `subject` 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 (s. Kapitel 3.3.4 "Name der Zertifizierungsstelle"). Auch für das `subject`-Feld sind nur X.500-Distinguished-Names zulässig.

Wie oben beschrieben, lassen sich im vorliegenden Profil gemäß Abschnitt 2.2 der Security Schnittstelle die beiden Typen von DNs "allgemeine Teilnehmerzertifikate" und „Zertifizierungsstellen (PCA oder CA)“ unterscheiden.

Für das Feld `subject` ist der Ertere der Typen relevant, die Zusammensetzung der Attribute für diesen Typ wird entsprechend der bereits oben angewendeten Darstellung in der folgenden Tabelle aufgeführt:

Aufbau des DNs im "subject"-Datenfeld für Teilnehmerzertifikate

Pos.	Attribute		Erläuterung
1	CountryName (verpflichtend)	C	Zweistelliges Kürzel für die Länderkennung, wie "DE" für Deutschland.
2	OrganizationName (verpflichtend, fest)	O	Name des Trust Centers als feste Zeichenkette - „ITSG TrustCenter fuer den Datenaustausch im Gesundheitswesen“
3	OrganizationUnitName (verpflichtend)	OU	Name der Institution (Firmenname des Leistungserbringers oder des Arbeitgebers)

⁵ Anmerkung: Nach dem Profil des NIST ist stets „UTCTime“ zu verwenden. Das Sigl-Profil sieht stets „GeneralizedTime“ vor. Die IETF schreibt vor, ab dem Jahr 2050 „GeneralizedTime“ zu verwenden, davor „UTCTime“. Entsprechend dem Ansatz des Sigl-Profiles soll so schnell wie möglich auf „GeneralizedTime“ umgestellt werden.

⁶ Das 'Z' am Ende steht für „Zulu-Zeit“ und ist die allgemein übliche Kurzform für Greenwich Mean Time (GMT).

⁷ Sekunden sind stets anzugeben, auch wenn die Anzahl der Sekunden Null ist.

Pos.	Attribute		Erläuterung
4	OrganizationUnitName (verpflichtend)	OU	Institutionskennzeichen oder Betriebs- bzw. Zahlstellennummer. Mit vorangestellter Kennung „IK“ (bei Leistungserbringer) oder „BN“ (bei Arbeitgeber oder Zahlstelle).
5	CommonName (verpflichtend)	CN	Der Name einer natürlichen Person, die als Ansprechpartner für die Institution fungiert.

Aufbau des DN's im "subject"-Datenfeld für Zertifizierungsstellen

Pos.	Attribute		Erläuterung
1	CountryName (verpflichtend)	C	Zweistelliges Kürzel für die Länderkennung, wie "DE" für Deutschland.
2	Organization Name (verpflichtend, fest)	O	Name des Trust Centers, das das Zertifikat ausstellt. Für beide Zertifikatsarten (PCA-Zertifikat und CA-Zertifikat) ist hier der Name der PCA einzutragen. Die hierfür festgelegten Zeichenketten siehe oben bzw. in Abschnitt 3.3.4.

Profilierung:

Die oben in der Tabelle ausgeführten Angaben zum DN sind verbindlich. Im Übrigen gelten die Anforderungen an DN's im Feld `issuer` entsprechend.

In welcher Reihenfolge die Attribute im Namen enthalten sind, ist für dieses Profil festgelegt. Die Reihenfolge der durch die Spalte "Pos." angegebenen Position der Attribute muss eingehalten werden.

Die von [IMTTP1] vorgesehene Pseudonyme-Regelung nach dem 31.12.2003 wird nicht unterstützt, da eine Pseudonymisierung des Zertifikatsinhabers nicht wünschenswert ist.

3.3.7 Basic Constraints

Das Feld `BasicConstraints` ist eine optionale Datenstruktur, die das Zertifikat des Teilnehmers einer Rolle zuordnet. Die Zertifikate der CA und PCA müssen mit `CA=TRUE` versorgt werden. Die Teilnehmerzertifikate müssen `CA=FALSE` versorgt werden.

Bei optionaler Verwendung gilt folgende Datenstruktur:

```
BasicConstraints ::= SEQUENCE {
    CA {TRUE, FALSE}
    pathLenConstraint OCTET STRING}
```

3.3.8 Öffentlicher Schlüssel des Zertifikatsinhabers

Referenz: [SecSchnitt], Abschnitte 2.1.6 und 3.2.7
[IMTTP1], Kapitel 2, Tabelle 2.#8 und 2.#14

Das Feld `subjectPublicKeyInfo` ist eine Datenstruktur, die den öffentlichen Schlüssel des Zertifikatsinhabers im Teilfeld `subjectPublicKey` und den Algorithmus, mit dem der Schlüssel zu verwenden ist, im Teilfeld `algorithm` enthält.

Es wird folgende Datenstruktur verwendet:

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

Die Struktur für den Datentyp `AlgorithmIdentifier` wurde bereits im Zusammenhang mit den Feldern `signatureAlgorithm` und `signature` erläutert (s. Kapitel 3.2 und 3.3.3).

In Kapitel 8 werden die in diesem Profil vorgesehenen `AlgorithmIdentifier` zusammen mit dem Zweck beschrieben, für den der jeweilige Algorithmus verwendet werden darf (Signieren, Verschlüsseln). Dort finden sich auch Vorgaben für die Kodierung der öffentlichen Schlüssel.

3.3.9 Nicht verwendete optionale Datenfelder

Eindeutige Bezeichner für CAs und Zertifikatsinhaber

Referenz: [SecSchnitt], Abschnitt 4.4.3
[IMTTP1], Kapitel 2, Tabelle 2.#8 und 2.#14

Profilierung:

Gemäß [IMTTP1, Tabelle 2.#9 und 2.#10] ist es verboten (`MUST NOT`), die optionalen Datenfelder `issuerUniqueId` und `subjectUniqueID` zu verwenden. Es ist erforderlich, dass die in den Datenfeldern `issuer` (vgl. Abschnitt 3.3.4) und `subject` (vgl. Abschnitt 3.3.6) eingetragenen DNs eindeutig sind.

4 Nachrichtenaustausch

Für die Ergänzung zur „Security Schnittstelle für das Gesundheitswesen Version 2.0.1“ sind insgesamt drei verschiedene Nachrichtentypen nach PKI-Nachrichten und Nachrichten für den Datenaustausch zu unterscheiden.

4.1 PKI-Nachrichten

PKI-Nachrichten ergeben sich aus den Kommunikationsschritten im Beantragungsverfahren für ein Zertifikat. Das Beantragungsverfahren für ein Zertifikat besteht aus zwei wesentlichen Kommunikationsschritten:

- Der Antragsteller, der ein Zertifikat wünscht, stellt einen Antrag bei der Zertifizierungsinstanz (CA-Certification Authority) (Typ 1: Zertifizierungsanfrage).
- Die Zertifizierungsinstanz schickt eine Antwort mit Zertifikat zurück (Typ 2: signierte Zertifizierungsantwort).

4.2 signierte und verschlüsselte Nachrichten

Nachrichten werden im Datenaustausch des Gesundheitswesens gesichert übertragen (Typ 3: signierte und verschlüsselte Nachricht). In der Antwort der Zertifizierungsinstanz ist das beantragte Zertifikat enthalten, mit dem der Teilnehmer (Leistungserbringer oder Arbeitgeber) seine Nachrichten signiert. Die Zertifikate haben eine definierte Gültigkeit, in der der signierte und verschlüsselte Datenaustausch mit den Krankenkassen durchgeführt werden kann. Der Nachrichtenaustausch findet zwischen den folgenden Beteiligten statt:

- Leistungserbringer und Krankenkasse zur Übermittlung der Abrechnungsdaten oder
- Arbeitsgebern zur Übermittlung der Versicherungsmeldungen zur Krankenkasse.

Zu den drei aufgeführten Nachrichtentypen lassen sich verschiedene Standards finden; in Ergänzung zur „Security Schnittstelle für das Gesundheits- und Sozialwesen“ werden die beiden von [SecSchnitt] und [IMTTP2] vorgesehenen Standards PKCS#10 (für Typ 1: Zertifizierungsanfrage) und PKCS#7 (für Typ 2 und 3: Zertifizierungsantwort und verschlüsselte Nachrichten) ausgeführt.

In Kapitel 5 werden die Anforderungen an eine Zertifizierungsanfrage nach PKCS#10 in den Ergänzung zur „Security Schnittstelle für das Gesundheitswesen Version 2.0.1“ beschrieben. Kapitel 6 beschreibt die Anforderungen an eine Zertifizierungsantwort und an die verschlüsselten Nachrichten.

5 PKCS#10-Zertifizierungsanfrage

5.1 Überblick

Eine Zertifizierungsanfrage nach PKCS#10 besteht aus

- einem Distinguished Name (DN),
- einem öffentlichen Schlüssel und
- einem Satz an Erweiterungen, welche zusammen vom Antragsteller (mit seinem zum öffentlichen Schlüssel gehörenden privaten Schlüssel) signiert werden.

Eine Anfrage ist eine selbstsignierte Datenstruktur (ein sogenanntes selbstsigniertes Zertifikat). Sie weist in großen Teilen eine zu einem X.509v3-Zertifikat vergleichbare Datenstruktur auf. Diese Datenstruktur ist an den Stellen reduziert, an denen eine Dateneinheit keinen Sinn hat. So ist beispielsweise kein `issuer`-Datenfeld zu finden, das im X.509v3-Zertifikat die ausstellende Instanz angibt.

Zertifizierungsanfragen werden zu einer Zertifizierungsinstanz (Certification Authority - CA) gesendet, die die Anfrage in ein X.509-Zertifikat transformiert. In welcher Form die CA das nun von ihr signierte Zertifikat zurückgibt, ist nicht Gegenstand des PKCS#10-Standards. Eine PKCS#7-Nachricht ist eine der möglichen Formen.

5.2 Aufbau des PKCS#10-Datentyps

Referenz: [SecSchnitt], Abschnitt 4.8
Der Datentyp eines PKCS#10-Request ist mit dem ASN.1-Datentyp `CertificationRequest` in [IMTTP2] Tabelle 1 und in [PKCS#10] in Abschnitt 4.2 festgelegt:

aus [IMTTP2] Tabelle 1:

1	<code>certificationRequestInfo</code>	DER-kodierte Anfrage-Information, bestehend aus:
1.1	<code>version</code>	Versionsnummer
1.2	<code>Subject</code>	DN des Antragstellers
1.3	<code>subjectPublicKeyInfo</code>	Informationen über den öffentlichen Schlüssel, der zu zertifizieren ist
1.4	<code>attributes</code>	Satz an Erweiterungen zum Feld "subject"
1.4.1	<code>ExtensionReq</code>	Erweiterung, die es erlaubt, eine oder mehr Attribute nach Standard-X.509v3-Erweiterungen hinzuzufügen
2	<code>signatureAlgorithm</code>	Kennzeichen für den Signaturalgorithmus, mit dem <code>CertificationRequestInfo</code> signiert wird.
2.1	<code>Signature</code>	Ergebnis des Signierens von <code>CertificationRequestInfo</code> , dargestellt als Datentyp BITSTRING

aus [PKCS#10] in Abschnitt 4.2:

```
CertificationRequest ::= SEQUENCE {
  certificationRequestInfo CertificationRequestInfo,
  signatureAlgorithm AlgorithmIdentifier{{ SignatureAlgorithms }},
  signature BIT STRING
}
```

Dieser Aufbau entspricht dem äußeren Aufbau eines X.509v3-Zertifikats, die weiter innen liegenden Datenstrukturen unterscheiden sich jedoch. Vor allen Dingen ist die Datenstruktur gegenüber der X.509v3-Zertifikatsstruktur um Felder reduziert, die für eine Zertifizierungsanfrage keine Bedeutung haben, so z. B. der Gültigkeitszeitraum `validity`.

`CertificationRequestInfo` ist die Anfrageinformation; dies ist das Datenfeld, das zu signieren ist.

Für `signatureAlgorithm` gelten die Angaben, die im Abschnitt 3.2 „Identifizierung des verwendeten Signaturalgorithmus“ zum `signatureAlgorithm` der Datenstruktur `TBSCertificate` gemacht worden sind. `signatureAlgorithm` kennzeichnet den Signaturalgorithmus, mit dem `CertificationRequestInfo` signiert wird.

Profilierung:

Die Security Schnittstelle legt fest, dass als Signaturalgorithmus SHA-1 bzw. SHA-256 mit RSA einzusetzen ist; vgl. Abschnitt 8.2. Das Feld `parameters` ist mit dem Wert NULL zu belegen.

Das Datenfeld `signature` enthält das Ergebnis des Signierens von `CertificationRequestInfo` mit dem privaten Schlüssel des Antragstellers, der im Datenfeld `CertificationRequestInfo.subject` angegeben ist.

5.3 Aufbau der Teildatenstruktur `CertificationRequestInfo`

Der Teildatenstruktur `CertificationRequestInfo` liegt folgende ASN.1-Struktur zugrunde:

aus [PKCS#10] in Abschnitt 4.1 (vgl. auch obige Tabelle aus [IMTTP2]) :

```
CertificationRequestInfo ::= SEQUENCE {
  version INTEGER { v1(0) } (v1,...),
  subject Name,
  subjectPKInfo SubjectPublicKeyInfo{{ PKInfoAlgorithms }},
  attributes [0] Attributes{{ CRIAttributes }}
}
```

Die Datenfelder `version`, `subject`, `subjectPKInfo` und `attributes` werden in den folgenden Abschnitten beschrieben.

5.3.1 Versionsnummer

Dieses Datenfeld entspricht nicht der Versionsnummer eines X.509v3-Zertifikats.

Profilierung:

In [IMTTP2], Tabelle 1 ist die Versionsnummer `v1(0)` für dieses Feld festgelegt, da in [PKCS#10] Abschnitt 4.1 verlangt wird: „It „ (- gemeint ist: `version` -) “shall be 0 for this version of the standard”.

5.3.2 Namen von Zertifikatsinhabern

Das Datenfeld `subject` in einem PKCS#10-Datentyp entspricht dem gleichnamigen Datenfeld in einem X.509v3-Zertifikat; vgl. Abschnitt 3.3.6 „Namen von Zertifikatsinhabern“.

Dieses Datenfeld soll exakt den Aufbau eines DNs für Teilnehmerzertifikate aufweisen, der für X.509v3-Zertifikate gefordert wird, denn die Angabe des DNs wird bei der Transformation in ein X.509v3-Zertifikat übernommen.

Profilierung:

Die Angaben in Abschnitt 3.3.6 „Namen von Zertifikatsinhabern“ zum DN für Teilnehmerzertifikate sind verbindlich.

5.3.3 Öffentlicher Schlüssel des Zertifikatsinhabers

Mit dem Datenfeld `subjectPublicKeyInfo` wird der öffentliche Schlüssel an die Zertifizierungsinstanz übergeben. Für dieses Datenfeld gelten dieselben Anforderungen, wie an das gleichnamige Datenfeld in der X-509v3-Zertifikatsstruktur. Dieses Feld wird bei der Transformation in ein X.509v3-Zertifikat übernommen.

Zusätzlich wird dieses Feld daraufhin überprüft, ob der hierin enthaltene öffentliche Schlüssel zu dem privaten Schlüssel gehört, der die Zertifizierungsanfrage signiert hat (engl. Proof of Possession – dt. Besitznachweis).

Die im Antrag enthaltene Signatur ist eine Signatur über die gesamte Datenstruktur `CertificationRequestInfo` und steht im Datenfeld `CertificationRequest.signature`. Diese Signatur wird gegen das Datenfeld `subjectPublicKeyInfo` geprüft.

Die Signatur ist gültig, wenn sie sich mit dem öffentlichen Schlüssel – also dem Datenfeld `subjectPublicKeyInfo` – verifizieren lässt. Damit ist es nicht erforderlich, den privaten Schlüssel bei der Antragstellung an die Zertifizierungsstelle zu übergeben, sondern der private Schlüssel verbleibt in sicherer Verwahrung.

5.3.4 Profilierung:

Die Angaben in Abschnitt 3.3.7 „Basic Constraints

Das Feld `BasicConstraints` ist eine optionale Datenstruktur, die das Zertifikat des Teilnehmers einer Rolle zuordnet. Die Zertifikate der CA und PCA müssen mit `CA=TRUE` versorgt werden. Die Teilnehmerzertifikate müssen mit `CA=FALSE` versorgt werden.

Bei optionaler Verwendung gilt folgende Datenstruktur:

```
BasicConstraints ::= SEQUENCE {  
    CA {TRUE, FALSE}  
    pathLenConstraint OCTET STRING}  
    Öffentlicher Schlüssel des Zertifikatsinhabers
```

Öffentlicher Schlüssel des Zertifikatsinhabers zum Aufbau des Datenfeldes `subjectPublicKeyInfo` ist verbindlich.

5.3.5 Erweiterungen

Da es nicht erforderlich ist, dass bei der Antragstellung Daten für die später in den Zertifikaten verwendeten Erweiterungen übergeben werden, sondern diese ausschließlich von der Zertifizierungsinstanz selbst gesetzt werden, sollen die Datenfelder `attributes`, insbesondere `ExtensionReq` leer sein.

Profilierung:

Die Zertifizierungsanfrage soll keine Erweiterungen enthalten.

5.4 Erklärung einer PKCS#10 Zertifizierungsanfrage gemäß Kapitel 5

Das hier gezeigte Beispiel einer PKCS#10 Zertifizierungsanfrage mit der Hash-Variante SHA-1 wird mit dem ASN.1 Anzeigetool BERViewer dargestellt und entsprechend kommentiert.

Die Zertifizierungsanfrage ist in drei Teile aufgegliedert, dies ist nur der Übersichtlichkeit geschuldet.

5.4.1 Teil 1, Version und Subject Name

```
3- SEQUENCE, Length = 721
  - SEQUENCE, Length = 441
    - INTEGER, Length = 1, Value = 0 (0x0)
    - SEQUENCE, Length = 139
      - SET, Length = 11
        - SEQUENCE, Length = 9
          - OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 6 } countryName
          - PrintableString, Length = 2, Value = "DE"
        - SET, Length = 58
          - SEQUENCE, Length = 56
            - OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 10 } organizationName
            - PrintableString, Length = 49, Value = "ITSG TrustCenter fuer sonstige Leistungserbringer"
          - SET, Length = 18
            - SEQUENCE, Length = 16
              - OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 11 } organizationalUnitName
              - PrintableString, Length = 9, Value = "ITSG GmbH"
            - SET, Length = 20
              - SEQUENCE, Length = 18
                - OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 11 } organizationalUnitName
                - PrintableString, Length = 11, Value = "IK660530056"
            - SET, Length = 22
              - SEQUENCE, Length = 20
                - OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 3 } commonName
                - PrintableString, Length = 13, Value = "Marcell Imhof"
```

Versionsnummer
Kapitel 5.3.1

Distinguished Name
Kapitel 5.3.2

5.4.2 Teil 2, PublicKeyInfo

```

SEQUENCE, Length = 290
├── SEQUENCE, Length = 13
│   ├── OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 1 } rsaEncryption ← AlgorithmIdentifier
│   └── NULL ← Feld Parameters mit Vorgabewert NULL
└── BIT STRING, Length = 271, Unused bits = 0
    1: 30 82 01 0A 02 82 01 01 00 80 0.....
    11: 01 C6 B5 27 5E C2 4B 03 2B C3 .F5'^BK.+C
    21: 8                                     |'E!.
    31: E                                     /b+.C.
    41: E                                     /oO...
    51: D                                     :*'IaW
    61: 6                                     \,,$43
    71: C                                     W;^.#
    81: 7                                     .{6z.
    91: 5                                     !@_!eY
    101: F                                    B|9q.
    111: D                                    f|i..
    121: 8                                    ..$.U
    131: 0                                    |E=I>=
    141: 2                                    D]}N:
    151: D                                    >9v.R|
    161: 2                                    |e.cg
    171: 9                                    |.>?V'
    181: 0                                    |.[:V
    191: E                                    |.ER..
    201: B                                    -k/Hj"
    211: 6                                    |v.p..
    221: 0                                    U[_.
    231: F                                    |.)zc5
    241: 3                                    ..$r.(
    251: 71 EC 88 DE 86 49 14 B4 E9 1D q1.^.I.4i.
    261: 86 8A A9 5C 03 02 03 01 00 01 ..)\.....
[0] Context-Specific, Length = 0
    
```

Schlüssel-
daten

PublicKey

PublicKeyInfo
Kapitel 5.3.3 und 8.1

5.4.3 Teil 3, CertificationRequest.Signature

```
[-] SEQUENCE, Length = 13
  ... OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 5 } sha1withRSAEncryption
  ... NULL
[-] BIT STRING, Length = 257, Unused bits = 0
  ... 1: 2A D2 59 F0 92 F9 88 25 75 10 *RYp.y.$u.
  ... 11: A1 53 95 0F 97 26 F9 90 0E 1F !S...$y...
  ... 21: E6 FF 86 82 B5 E2 24 E9 6B E1 f...5b$ika
  ... 31: EC D0 72 4B 18 05 33 61 05 89 lPrK..3a..
  ... 41: 72 05 84 DE 6E 38 75 83 C7 7A r..^n8u.Gz
  ... 51: 76 79 A8 7C 9A 17 96 E6 47 B8 vy(|...fG8
  ... 61: D8 9F 1D CB F0 CD 12 EF 1E B5 X..KpM.o.5
  ... 71: 47 4C 89 60 65 B5 F4 E2 57 32 GL.`e5tbW2
  ... 81: E2 4E 8E F3 1D 71 41 D4 54 04 bN.s.qATT.
  ... 91: AD 22 23 B4 93 0D D2 12 CD D5 -"#4..R.MU
  ...101: 80 77 0A B9 DE 97 D1 85 0C E2 .w.9^.Q..b
  ...111: E5 13 36 17 F0 63 CD F6 79 E7 e.6.pcmvvyg
  ...121: B9 1A B2 61 1B 5E D2 9E 02 41 9.2a.^R..A
  ...131: 20 AA 92 D6 8A 20 D7 F2 E9 88 *.V. Wri.
  ...141: 24 EF 5D F9 A4 05 C9 60 DB 8A $o]y$.I`[.
  ...151: 68 DF 17 F3 83 D5 E1 14 F5 5B h_.s.Ua.u[
  ...161: E1 27 86 85 4B D9 99 1C B1 51 a'..KY..1Q
  ...171: B8 4C 3C 52 16 06 D4 08 F6 A3 8L<R..T.v#
  ...181: E2 CA CC DD 54 ED FE F1 0C 2C bJL]Tm~q.,
  ...191: FA 3A 30 7F 35 19 F6 6F 68 79 z:0.5.vohy
  ...201: 3A 58 75 E9 DD D7 5B F3 33 D0 :Xui]W[s3P
  ...211: F5 1B 7D B3 E2 E6 01 2C A0 B0 u.)3bf., 0
  ...221: 6A 1D D4 68 98 07 B9 1E D9 35 j.Th..9.Y5
  ...231: 00 6F 1F 37 F4 11 C6 3F 58 D2 .o.7t.F?XR
  ...241: 53 D3 25 D1 FD 06 AC 3F F0 8F SS$Q}.,?p.
  ...251: 06 43 BB 6B 03 1B .C;k..
```

AlgorithmIdentifier

Feld Parameters mit Vorgabewert NULL

Signatur

CertificationRequest.Signature
Kapitel 3.1 und 3.2

5.5 Transport der PKCS#10 Zertifizierungsanfrage

5.5.1 Transportformat

Referenz: [IMTTP2], Abschnitt 2.3,
[SecSchnitt] Abschnitte 2.3 und 4.8

Nachdem ein Datenobjekt vom Typ `certificationRequestInfo` (vgl. 5.2 „Aufbau des PKCS#10-Datentyps“) so zusammengestellt wurde, wie von der Datentyp-Spezifikation definiert, liegt es als ASN.1-Struktur vor. Es folgt eine DER-Kodierung, die als Transportsicherung dient. Der daraus resultierende Bytestream wird in eine Datei abgelegt, die die Dateiendung „p10“ aufweist. Das physikalisch so gesicherte PKCS#10-Datenobjekt kann auf verschiedenen Transportwegen zur CA gesendet werden.

Physikalisch handelt es sich bei einer PKCS#10-Datei um eine Binärdatei, die auf den weiteren Transportwegen als eben solche zu behandeln ist.

5.5.2 Transportwege

Da der Transport von Dateien über das Internet über eine Vielzahl an Protokollen, wie https, http, ftp, etc. oder auch per E-Mail erfolgen kann, wird vorgeschlagen, hierfür keine weiteren „Verpackungen“, sogenannte Umschläge zu verwenden.

Die zu berücksichtigenden Transportwege zwischen den Kommunikationspartnern des ITSG-Trust Centers zum Austausch von Zertifizierungsanfragen sind:

- Transport per Diskette oder
- E-Mail an einen automatisierten Mail-Empfänger.

Die folgende Tabelle legt Details zum Transport auf dem jeweiligen Transportweg fest.

Transportweg	Details
Diskette	„p10“-Datei mit der PKCS#10-Datenstruktur wird als Binärdatei auf der obersten Verzeichnisebene abgelegt. Die Diskette ist im DOS-Format (FAT) formatiert.
E-Mail	„p10“-Datei mit der PKCS#10-Datenstruktur wird als Anhang an eine Mail versendet.

Weitere Transportwege sind in den Richtlinien zum Datenaustausch festgelegt.

Es muss beachtet werden, dass jedes Trust Center nur bestimmte Transportwege anbietet, unabhängig der möglichen Transportwege in den Richtlinien zum Datenaustausch. Sehen Sie dazu in den Webseiten der Trust Center (z. B. www.trustcenter.info)

6 PKCS#7-Zertifizierungsantwort

Referenz: [IMTTP2], Abschnitt 2.3,
[SecSchnitt] Abschnitte 2.1.8 und 2.3

6.1 Überblick PKCS#7

In welcher Form die CA das von ihr signierte Zertifikat zurückgibt, ist unter anderem Gegenstand von PKCS#7. PKCS#7 legt Anforderungen für viele Nachrichtentypen (vgl. nachfolgende Liste in diesem Abschnitt mit den verschiedenen Nachrichten-Typen) fest, z. B. eine PKCS#7-Zertifizierungsantwort.

PKCS#7⁸ wird auch als Cryptographic Message Syntax Standard (CMS) bezeichnet und beschreibt eine Syntax, nach der Daten durch kryptographische Maßnahmen wie digitale Signaturen oder Verschlüsselung geschützt werden können

Die allgemeine Syntax eines CMS-Objektes ist:

```
ContentInfo ::= SEQUENCE {  
    contentType ContentType,  
    content  
    [0] EXPLICIT ANY DEFINED BY contentType OPTIONAL }  
ContentType ::= OBJECT IDENTIFIER
```

Im Feld „contentType“ wird der Typ des geschützten Objektes durch einen OID angegeben. Im Feld „content“ sind die geschützten Daten enthalten.

Insgesamt sind für CMS-Objekte sechs „Inhalts“-Typen (die `content types`) definiert.

Jeder der Typen zeichnet sich durch die Verfahren aus, die auf die ihm anvertrauten Daten - den Inhalt - anzuwenden sind, um eine besondere Form von Schutz zu gewährleisten. Es wird zwischen Basistypen und erweiterten Typen unterschieden. Basistypen haben keine kryptographische Funktionalität. Die folgende Tabelle gibt einen kurzen Überblick über die Typen:

Typ	Typenklasse	Bedeutung
Data	Basistyp	Modellierung von Daten
SignedData	erweiterter Typ	Format, um Datenintegrität und Senderauthentizität durch die Verwendung von digitalen Signaturen und Zertifikaten zu gewährleisten.
EnvelopedData	erweiterter Typ	Empfängerspezifische Verschlüsselung von Daten
SignedAnd- EnvelopedData	erweiterter Typ	Kombination von Signed-data und Enveloped-data: digitale Signatur und Verschlüsselung
DigestedData	erweiterter Typ	Gewährleistung der Integrität von Daten durch einen Hashwert
EncryptedData	erweiterter Typ	Datenverschlüsselung

⁸ PKCS#7v1.5 wird als CMS bezeichnet. Es gibt von PKCS#7 bereits eine Nachfolgerversion, die jedoch nicht verwendet werden soll. PKCS#7v1.5 dient als Grundlage für S/MIME Version 3.

6.2 Aufbau der PKCS#7-Zertifizierungsantwort

Eine PKCS#7-Zertifizierungsantwort nach [IMTTP2] ist ein CMS-Datenobjekt vom „Inhalts“-Typ `SignedData`.

Profilierung:

[IMTTP2] fordert für die PKCS#7-Zertifizierungsantwort den Einsatz des `content-type signedData` mit der OID 1.2.840.113549.1.7.2

CMS-Objekte vom Typ `SignedData` umfassen die zu schützenden Daten und eine oder mehrere digitale Signaturen. Sie werden üblicherweise durch die folgenden Schritte generiert:

1. Es wird ein Hash-Wert über die zu schützenden Daten gebildet.
2. Die Signatur wird durch Anwendung des privaten Signaturschlüssels auf den Hash-Wert gebildet.
3. Jede Signatur wird mit anderen für die Signatur spezifischen Werten zu einem Wert vom Typ `SignerInfo` zusammengefasst.
4. Der Hash-Algorithmus wird mit dem Wert `SignerInfo` und den zu schützenden Daten zu einem Wert vom Typ `SignedData` zusammengefasst.
5. Da es sich bei einer Zertifizierungsantwort um ein sogenanntes **degeneriertes SignedData** handelt, gelten Einschränkungen, die im Folgenden bei der Beschreibung der einzelnen Datenfelder erläutert werden. Gemäß [IMTTP2], Abschnitt 2.1.1 entfallen die Datenfelder `encapContent` und `signerInfos` für eine Zertifizierungsantwort aus dem CMS-Datenobjekt.
6. Der Zusatz „degeneriert“ zu `SignedData` bezieht sich auf die Sonderrolle des `SignedData` als Übertragungsobjekt für ein neues von der CA ausgestelltes Zertifikat zum Teilnehmer. Eigentlich wird ein `SignedData` verwendet, um signierte Daten zusammen mit dem Zertifikat, dessen Inhaber mit seinem privaten Schlüssel signiert hat, zu transportieren. In diesem Fall kommt es überhaupt nicht auf den Dateninhalt an, sondern nur auf das mitgelieferte Zertifikat, das der Empfänger das erste Mal erhält. Zur Übergabe an den Empfänger wird es in ein `SignedData` gepackt.
7. Die Struktur der PKCS#7-Zertifizierungsantwort wird in [IMTTP2], Abschnitt 2.1.2. und [PKCS#7], Kapitel 7 durch den ASN.1-Datentyp `ContentInfo` (siehe oben) und in [PKCS#7], Abschnitt 9.1 durch den festgeschriebenen `ContentType SignedData` spezifiziert:

aus [PKCS#7] Abschnitt 9.1:

```
SignedData ::= SEQUENCE {
    version Version,
    digestAlgorithms DigestAlgorithmIdentifiers,
    contentInfo ContentInfo,
    certificates
        [0] IMPLICIT ExtendedCertificatesAndCertificates
        OPTIONAL,
    crls
        [1] IMPLICIT CertificateRevocationLists OPTIONAL,
    signerInfos SignerInfos }
```

aus [IMTTP2] Tabelle 1:

1	certificationRequestInfo	DER-kodierte Anfrage-Information, bestehend aus:
1.1	version	Versionsnummer
1	ContentType	Indication of the type of content
2	Content	Content of signed-data
2.1	Version	Version number of CMS syntax
2.2	DigestAlgorithms	Collection (including zero) of message digest algorithm identifiers
2.3	EncapContentInfo contentInfo	Data to be protected
2.4	Certificates	Collection of certificates
2.5	Crls	Collection of CRLs
2.6	SignerInfos	Collection of per-signer information

In den nachfolgenden Abschnitten werden die Anforderungen an die Datenfelder der Struktur `SignedData` aufgeführt.

6.3 Aufbau der Teildatenstruktur Content vom ContentType „SignedData“

6.3.1 Datenfeld „version“

Das Feld `version` enthält die Versionsnummer der Syntax. Die Versionsnummer ist je nach Art der zu signierenden Daten entweder „1“ oder „3“⁹.

Profilierung:

Für das Datenfeld `version` soll in dieser Datenstruktur immer der Wert „1“ eingesetzt werden.

6.3.2 Datenfeld „digestAlgorithms“

Dieses Datenfeld enthält genau einen Objektbezeichner¹⁰ für den Hash-Algorithmus, der zur Signatur von `content` eingesetzt wird.

Der zu verwendende Objektbezeichner ist von [IMTTP2] Tabelle 2 auf OID 1.3.14.3.2.26 für SHA-1 bzw. auf OID 2.16.840.1.101.3.4.2.1 für SHA-256 festgelegt; vgl. Abschnitt 8.1.

6.3.3 Datenfeld „encapContentInfo“

Das Datenfeld `encapContentInfo` enthält üblicherweise in einer `SignedData`-Struktur die zu schützenden Daten.

Profilierung:

Gemäß [IMTTP2], Abschnitt 2.1.1 entfallen die Datenfelder `encapContent` und `signerInfos` für eine Zertifizierungsantwort aus dem CMS-Datenobjekt.

⁹ Dies entspricht den Versionsnummern aus Kapitel 5.1 [RFC 2630 99]. Die Versionsnummer 1 ist zu verwenden, wenn uninterpretierte binäre Daten (OID „id-data“) signiert werden sollen. Falls den Daten jedoch ein Formatbezeichner zugewiesen ist (in der vorliegenden Spezifikation werden OIDs für die verschiedenen Formatbezeichner verwendet) muss die Versionsnummer „3“ sein.

¹⁰ Theoretisch sind mehrere OID-Angaben möglich, aber in diesem Zusammenhang besitzt das PKCS#10-Objekt nur eine Signatur für `SignedData`, und daher nur eine Angabe des hierfür verwendeten Hash-Algorithmus.

6.3.4 Datenfeld “certificates”

Dieses Datenfeld enthält das neu von der Zertifizierungsinstanz erstellte Zertifikat und alle Zertifikate des Zertifizierungspfades in der folgenden Form:

```
ExtendedCertificatesAndCertificates ::= SET OF ExtendedCertificateOrCertificate

ExtendedCertificateOrCertificate ::= CHOICE {
    certificate Certificate, -- X.509
    extendedCertificate [0] IMPLICIT ExtendedCertificate
}
```

Die Reihenfolge ist in [IMTTP2] nicht festgelegt. Das Datenfeld `certificates` besteht also aus einer nicht geordneten Liste von (X.509-)Zertifikaten oder erweiterten Zertifikaten (`ExtendedCertificates`)

Profilierung:

Erweiterte Zertifikate werden in diesem Profil nicht verwendet. Bei dem Datenfeld `certificates` handelt es sich daher um eine ungeordnete Liste von (X.509-)Zertifikaten.

6.3.5 Datenfeld “crls”

Das Feld `crls` ermöglicht es üblicherweise, dem Empfänger der Nachricht die Sperrlisten bereitzustellen, die er für die Verifikation der digitalen Signatur benötigt.

Profilierung:

Dieses Datenfeld wird in diesem Profil nicht verwendet und soll den Wert “Null” enthalten. Es ist geplant Sperrlisten in einem Verzeichnis zur Verfügung zu stellen.

6.3.6 Datenfeld “signerInfos”

Das Feld `signerInfos` enthält üblicherweise die Informationen über die Signierer, u. a. deren Signaturen.

Profilierung:

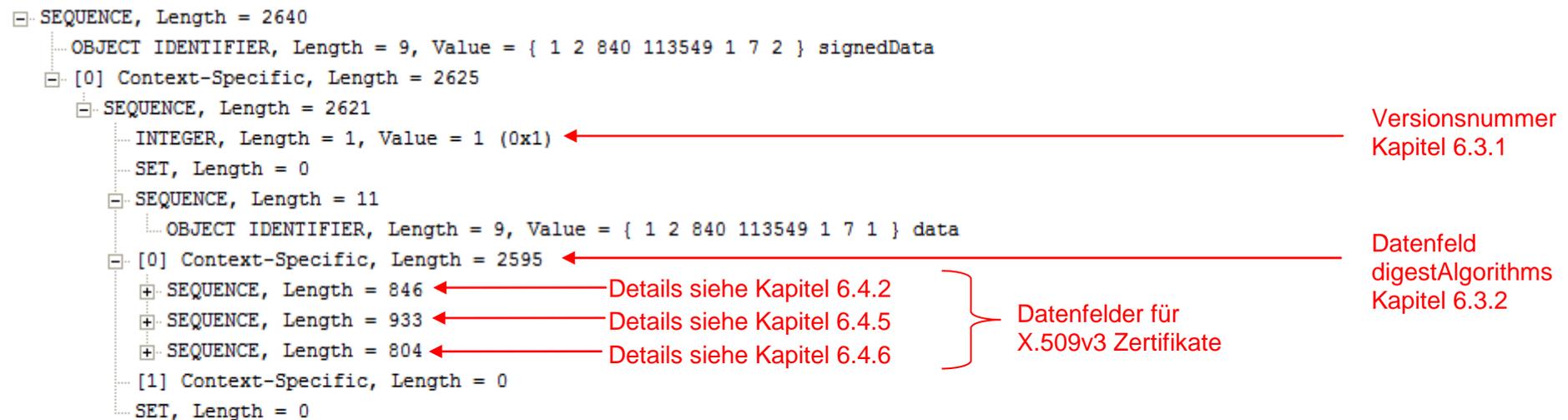
Gemäß [IMTTP2], Abschnitt 2.1.1 entfallen die Datenfelder `encapContent` und `signerInfos` für eine Zertifizierungsantwort aus dem CMS-Datenobjekt.

6.4 Erklärung einer PKCS#7 Zertifizierungsantwort gemäß Kapitel 6

Das hier gezeigte Beispiel einer PKCS#7 Zertifizierungsantwort mit der Hash-Variante SHA-1 wird mit dem ASN.1 Anzeigetool BERViewer dargestellt und entsprechend kommentiert.

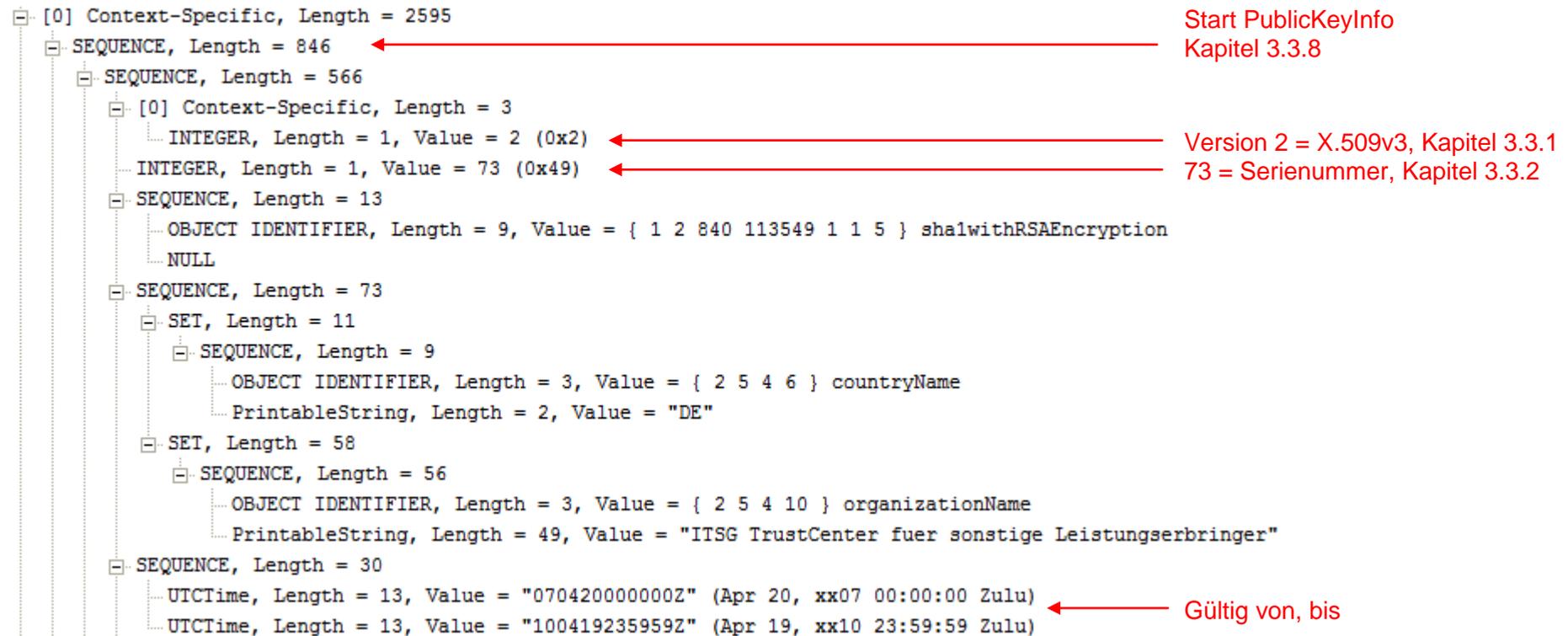
Die Zertifizierungsantwort ist in mehrere Teile aufgegliedert, um eine bessere Übersicht zu erhalten.

6.4.1 Teil 1, Version und Subject Name



6.4.2 Teil 2a, PublicKeyInfo

Dieser Teil zeigt die erste Sequence vom Teil 1 im Detail, sie enthält die Informationen zum Originator Zertifikat.



6.4.3 Teil 2b, PublicKeyInfo

Weitere Details des Originator Zertifikats...

```
[-] SEQUENCE, Length = 139
  [-] SET, Length = 11
    [-] SEQUENCE, Length = 9
      ... OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 6 } countryName
      ... PrintableString, Length = 2, Value = "DE"
    [-] SET, Length = 58
      [-] SEQUENCE, Length = 56
        ... OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 10 } organizationName
        ... PrintableString, Length = 49, Value = "ITSG TrustCenter fuer sonstige Leistungserbringer"
      [-] SET, Length = 18
        [-] SEQUENCE, Length = 16
          ... OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 11 } organizationalUnitName
          ... PrintableString, Length = 9, Value = "ITSG GmbH"
        [-] SET, Length = 20
          [-] SEQUENCE, Length = 18
            ... OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 11 } organizationalUnitName
            ... PrintableString, Length = 11, Value = "IK660530056"
          [-] SET, Length = 22
            [-] SEQUENCE, Length = 20
              ... OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 3 } commonName
              ... PrintableString, Length = 13, Value = "Marcell Imhof"
      [-] SEQUENCE, Length = 290
        [-] SEQUENCE, Length = 13
          ... OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 1 } rsaEncryption
          ... NULL
        [-] BIT STRING, Length = 271, Unused bits = 0
          1: 30 82 01 0A 02 82 01 01 00 80 0.....
```

Distinguished Name
Kapitel 5.3.2

PublicKey

6.4.4 Teil 2c, PublicKeyInfo

Weitere Details des Originator Zertifikats...

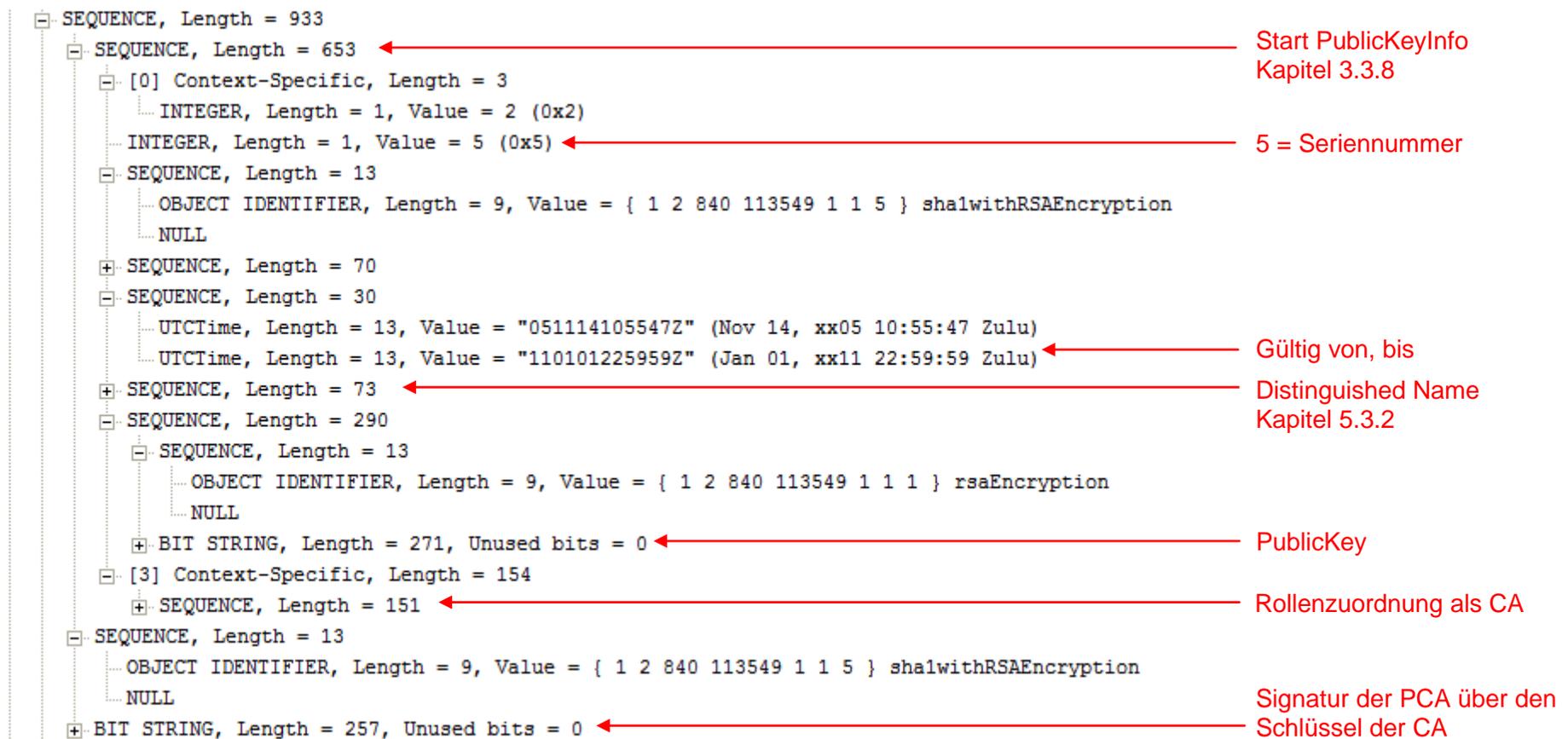
```
[-] SEQUENCE, Length = 13
  [...] OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 5 } sha1withRSAEncryption
  [...] NULL
[-] BIT STRING, Length = 257, Unused bits = 0
  [...] 1: 7E EB 2D 21 76 BE 5E CF 3B 4D ~k-!v>^O;M
  [...] 11: A9 6E A3 47 8E C5 5C 5D 63 5A )n#G.E\]cZ
  [...] 21: 3D E2 40 8A 97 96 06 C8 D9 D3 =b@...HYS
  [...] 31: B4 D1 D3 B1 81 14 F8 6A A8 D1 4QS1..xj(Q
  [...] 41: 59 38 D4 3F 49 2D 1D 3A 10 5E Y8T?I-..:^
  [...] 51: EE 65 07 7E 16 40 4C CC 72 79 ne.~.@LLry
  [...] 61: 2A B0 4E E3 CE 82 B3 78 08 9A *ONcN.3x..
  [...] 71: 96 A4 2A 07 24 55 13 F7 36 F7 .$.$.U.w6w
```

Signatur der CA über den
Schlüssel des Originator

6.4.5 Teil 3, IssuerKeyInfo

Dieser Teil zeigt die zweite Sequence vom Teil 1 im Detail, sie enthält die Informationen vom Issuer Zertifikat.

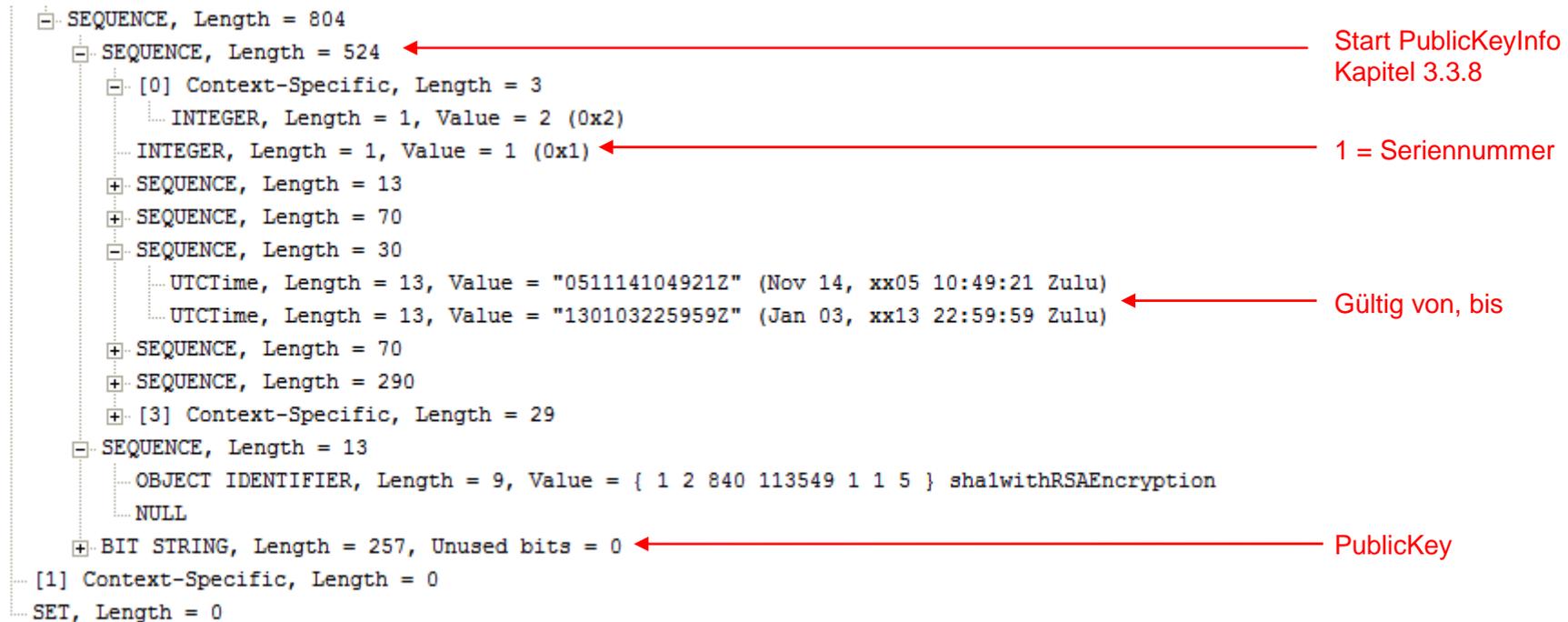
Da die Informationen stark dem oben gezeigten ähneln, wurde nicht jedes Detail beschrieben.



6.4.6 Teil 4, PCAKeyInfo

Der Vollständigkeit halber wird hier die dritte Sequence vom Teil 1 aufgeführt, sie enthält die Informationen des PCA Zertifikats.

Da die Informationen stark dem oben gezeigten ähneln, wurde nicht jedes Detail beschrieben.



6.5 Transport der PKCS#7-Zertifizierungsantwort

Nachdem für die PKCS#7-Nachrichten im Falle der signierten Nachricht mit einem Datenobjekt vom Typ `SignedData` erstellt wurde, liegt ein solches Datenobjekt als ASN.1-Struktur vor. Es folgt eine DER-Kodierung, die als Transportsicherung dient. Der daraus resultierende Bytestream wird in eine Datei abgelegt, die folgende Dateiendung aufweist:

Nachrichtenart	Dateiendung
Zertifizierungsantwort	p7c

6.5.1 Transportformat

Referenz: [IMTTP2], Abschnitt 2.3,
[SecSchnitt] Abschnitte 2.3 und 4.8

Eine Zertifizierungsantwort als PKCS#7-Datenobjekt wird in einer Datei abgelegt, die die Dateiendung „p7c“ aufweist. Das physikalisch so gesicherte Datenobjekt kann auf verschiedenen Transportwegen von der CA zum Teilnehmer gesendet werden.

Physikalisch handelt es sich bei einer „p7c“-Datei um eine Binärdatei, die auf den weiteren Transportwegen als eben solche zu behandeln ist.

6.5.2 Transportwege

Analog zum Transport der Zertifizierungsanfrage ohne Einbettung in einen MIME-Typ wird die Zertifizierungsantwort ebenfalls ohne digitalen Umschlag versendet.

Zum Transport von PKCS#7-Zertifizierungsantwort zu den Antragstellern geht man von einer möglichen Nutzung folgender Transportwege aus:

- Transport per Diskette,
- Versand per E-Mail und
- E-Mail-Responder, auf dem die Zertifizierungsantworten im Dateiformat zum Abruf bereitgestellt werden.

Das LDAP-Verzeichnis wird im Kapitel 9 ausführlich behandelt und soll hier nicht berücksichtigt werden.

Die folgende Tabelle legt Details zum Transport auf dem jeweiligen Transportweg fest.

Transportweg	Details
Diskette	„p7c“-Datei mit der PKCS#7-Datenstruktur für eine Zertifizierungsantwort wird als Binärdatei auf der obersten Verzeichnisebene abgelegt. Die Diskette ist im DOS-Format (FAT) formatiert.
E-Mail	„p7c“-Datei mit der PKCS#7-Datenstruktur für eine Zertifizierungsantwort wird als Anhang an eine Mail versendet.

7 PKCS#7-signierte und verschlüsselte Nachricht

Für den Datenaustausch zwischen Krankenkassen und Leistungserbringern / Arbeitgebern (unter Anwendung des von der CA erhaltenen Zertifikats, das sich in der Zertifizierungsantwort befindet,) dürfen beim PKCS#7-Verfahren ausschließlich **signierte und verschlüsselte Nachrichten** nach PKCS#7 eingesetzt werden.

Profilierung:

Für die signierte Nachricht wird der `ContentType SignedData` verwendet. Für die Verschlüsselung der signierten Nachricht wird der `ContentType EnvelopedData` verwendet.

Die PKCS#7-signierte und verschlüsselte Nachricht wird in [IMTTP3] behandelt. Abschnitt 4 beschreibt hierzu ein allgemeines Vorgehen (4.1 File Signature, 4.2 File Encryption). Abschnitt 3.2 und 3.3 liefern die erforderlichen Datenstrukturen.

7.1 Aufbau der PKCS#7-signierten und verschlüsselten Nachricht

Referenz: [IMTTP3] Tabelle 3.2
[IMTTP3] Tabelle 3.3,
[RFC 2630] Abschnitt 5
[RFC 2630] Abschnitt 6

Eine verschlüsselte Nachricht ist gemäß [IMTTP3] Abschnitt 3.3 vom Typ `EnvelopedData`, eine signierte gemäß [IMTTP3] Abschnitt 3.2 vom Typ `SignedData`. Die zu sichernden Daten werden wie in [IMTTP3] Abschnitt 4 beschrieben zuerst signiert und danach verschlüsselt.

Der Typ `SignedData` besteht allgemein aus den zu signierenden Daten, den für die Verifizierung der Signatur notwendigen Zertifikaten sowie Informationen zu dem signierenden Absender. Die Daten vom Typ `SignedData` werden wie in 7.2 beschrieben erstellt. Da hier jedoch das ganze `SignedData`-Objekt statt der degenerierten Form benötigt wird, sind folgende zusätzliche Felder zu verwenden:

- Die zu sichernden Daten werden im Feld `encapContentInfo` abgelegt.
- Die notwendigen Informationen zum Absender werden in ein Feld `SignerInfo` gefüllt.
- Die Signatur mit dem privaten Schlüssel des Absenders erfolgt wie gehabt.

Bei der Erstellung der Daten vom Typ `EnvelopedData`, gemäß Kapitel 7.4, wird das `SignedData`-Objekt als Input verwendet. Der Typ `EnvelopedData` besteht allgemein aus den zu verschlüsselnden Daten und einem für einen oder mehrere Empfänger verschlüsselten Nachrichtenschlüssel. Die Daten vom Typ `EnvelopedData` werden durch die folgenden Schritte generiert:

- Ein Nachrichtenschlüssel (content-encryption key) wird zufällig erzeugt.
- Die Daten, in diesem Fall das `SignedData`-Objekt, werden mit dem Nachrichtenschlüssel verschlüsselt.
- Der Nachrichtenschlüssel wird für jeden Empfänger mit dessen öffentlichem Verschlüsselungsschlüssel verschlüsselt.
- Der verschlüsselte Nachrichtenschlüssel und andere empängerspezifische Informationen werden in einem `RecipientInfo`-Wert zusammengefasst.
- Abschließend werden alle `RecipientInfo`-Werte mit den verschlüsselten Daten zum CMS-Objekt `EnvelopedData` zusammengesetzt.

7.2 Aufbau der Teildatenstruktur Content vom ContentType „SignedData“

Der Typ `SignedData` hat folgende Syntax:

```
SignedData ::= SEQUENCE {  
    version CMSVersion,  
    digestAlgorithms DigestAlgorithmIdentifiers,  
    encapContentInfo EncapsulatedContentInfo,  
    certificates [0] IMPLICIT CertificateSet OPTIONAL,  
    crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,  
    signerInfos SignerInfos }
```

7.2.1 Datenfeld „version“

Das Feld `version` enthält die Versionsnummer der Syntax.

Profilierung:

Für das Datenfeld `version` soll in dieser Datenstruktur immer der Wert „1“ eingesetzt werden.

7.2.2 Datenfeld „digestAlgorithms“

Dieses Datenfeld enthält genau einen Objektbezeichner¹¹ für den Hash-Algorithmus, der zur Signatur der Daten eingesetzt wird.

Als Hash-Algorithmen sind nur SHA-1 (auslaufend) und SHA-256 vorgesehen. Der zu verwendende Objektbezeichner ist von [IMTTP2] Tabelle 2 auf OID 1.3.14.3.2.26 (SHA-1) bzw. OID 2.16.840.1.101.3.4.2.1 (SHA-256) festgelegt; vgl. Abschnitt 8.1.

7.2.3 Datenfeld „encapContentInfo“

Das Datenfeld `encapContentInfo` enthält die zu signierenden Daten.

```
EncapsulatedContentInfo ::= SEQUENCE {  
    eContentType ContentType,  
    eContent [0] EXPLICIT OCTET STRING OPTIONAL }
```

Profilierung:

Gemäß [IMMTP3] wird hierfür der `eContentType` `id-data` mit der OID 1.2.840.113549.1.7.1 verwendet, die angibt, dass nicht interpretierte binäre Daten in das Teildatenfeld `eContent` eingetragen wurden.

7.2.4 Datenfeld „certificates“

Dieses Datenfeld enthält das Zertifikat, das für die Signatur verwendet wurde und alle dazugehörigen Zertifikate des Zertifizierungspfades:

```
CertificateSet ::= SET OF CertificateChoices  
  
CertificateChoices ::= CHOICE {  
    certificate Certificate, -- See X.509  
    extendedCertificate [0] IMPLICIT ExtendedCertificate, -- Obsolete  
    attrCert [1] IMPLICIT AttributeCertificate } -- See X.509 & X9.57
```

¹¹ Theoretisch sind mehrere OID-Angaben möglich, aber in diesem Zusammenhang besitzt das `SignedData`-Objekt nur eine Signatur, und daher nur eine Angabe des hierfür verwendeten Hash-Algorithmus.

Profilierung:

Verwendet werden ausschließlich X509-Zertifikate. Die Reihenfolge ist in [IMTTP2] nicht festgelegt und ist deshalb als Festlegungsvorschlag anzusehen.

Hier findet die Spezifikation der X509v3-Zertifikate Anwendung; vgl. Kapitel 3. Jedes der im PKCS#7-Datenobjekt unter der Datenstruktur `SignedData` im Feld `Certificates` anzugebende Zertifikat hat den im Kapitel 3 formulierten Anforderungen einschließlich der Profilierung an X509v3-Zertifikate zu genügen.

7.2.5 Datenfeld „crls“

Das Feld `crls` ermöglicht es üblicherweise, dem Empfänger der Nachricht die Sperrlisten bereitzustellen, die er für die Verifikation der digitalen Signatur benötigt.

Profilierung:

Dieses Datenfeld wird nicht verwendet und entfällt daher. Es ist geplant Sperrlisten in einem Verzeichnis zur Verfügung zu stellen.

7.2.6 Datenfeld „signerInfos“

Das Feld `signerInfos` enthält üblicherweise die Informationen über die Signierer, u. a. deren Signaturen.

Das Datenfeld hat folgende Syntax:

`SignerInfos ::= SET OF SignerInfo`

```
SignerInfo ::= SEQUENCE {
    version CMSVersion,
    sid SignerIdentifier,
    digestAlgorithm DigestAlgorithmIdentifier,
    signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL,
    signatureAlgorithm SignatureAlgorithmIdentifier,
    signature SignatureValue,
    unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL }
```

7.2.6.1 Datenfeld „version“

Das Feld `version` enthält die Versionsnummer der Syntax. Die Versionsnummer ist „1“.

Profilierung:

Für das Datenfeld `version` soll in dieser Datenstruktur immer der Wert „1“ eingesetzt werden.

7.2.6.2 Datenfeld „sid“

Das Feld `sid` enthält Hinweise um das Zertifikat des Absenders zu identifizieren.

```
SignerIdentifier ::= CHOICE {
    issuerAndSerialNumber IssuerAndSerialNumber,
    subjectKeyIdentifier [0] SubjectKeyIdentifier }
```

Profilierung:

Wie in [IMMTP3] empfohlen soll das Feld `IssuerAndSerialNumber` genutzt werden.

7.2.6.3 Datenfeld „digestAlgorithm“

Das Feld `digestAlgorithm` enthält den Identifikator des Hash-Algorithmus des Absenders.

Profilierung:

Siehe 8.1 Einweg-Hash-Funktionen (One-Way Hash Functions).

7.2.6.4 Datenfeld „signedAttrs“

Das Feld `signerAttrs` enthält zusätzliche Informationen, wie beispielsweise den Zeitpunkt der Signatur, die in die Signatur einbezogen werden.

Profilierung:

Die Verwendung dieses Feldes ist optional. Einzelheiten sind in [RFC 2630] beschrieben.

7.2.6.5 Datenfeld „signatureAlgorithm“

Das Feld `signatureAlgorithm` enthält den Identifikator des Signatur Algorithmus des Absenders.

Profilierung:

Siehe 8.2 Signaturalgorithmen (Signature Algorithms).

7.2.6.6 Datenfeld „signature“

Das Feld `signature` enthält die digitale Signatur des Absenders.

7.2.6.7 Datenfeld „unsignedAttrs“

Das Feld `unsignedAttrs` enthält zusätzliche Informationen, die nicht in die Signatur einbezogen werden.

Profilierung:

Dieses Feld entfällt.

7.3 Erklärung einer signierten Nachricht gemäß Kapitel 7.2

Das hier gezeigte Beispiel einer PKCS#7-signierten Nachricht mit der Hash-Variante SHA-1 wird mit dem ASN.1 Anzeigetool dargestellt und kommentiert. Im ersten Teil wird die signierte Datei beschrieben, im zweiten Teil folgen die Signaturinformationen der signierten Nachricht des Datenfeldes `signerInfos`.

7.3.1 Teil 1, die signierende Datei

- SEQUENCE, Length = Indefinite Length (4864)
 - OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 7 2 } signedData
 - [0] Context-Specific, Length = Indefinite Length (4851)
 - SEQUENCE, Length = Indefinite Length (4849)
 - INTEGER, Length = 1, Value = 1 (0x1) ← Versionsnummer Kapitel 7.2.1
 - SET, Length = 11
 - SEQUENCE, Length = 9
 - OBJECT IDENTIFIER, Length = 5, Value = { 1 3 14 3 2 6 } sha1 ← digestAlgorithm Kapitel 7.2.2
 - NULL
 - SEQUENCE, Length = 4460
 - OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 11354 9 1 7 1 } data
 - [0] Context-Specific, Length = 4445
 - OCTET STRING, Length = 4441 ← Das Datenfeld „encapContentInfo“ enthält die zu signierenden Daten Kapitel 7.2.3
 - | 1: 56 4F 53 5A 42 57 4E 41 43 33 VOSZBWNAC3 ← Das Beispiel zeigt eine Beitragsnachweis-Datei (Beginn des Vorlaufsatzes ...)
 - |

7.3.2 Teil 2, signerInfos

- SET, Length = 359
 - SEQUENCE, Length = 355
 - INTEGER, Length = 1, Value = 1 (0x1)
 - SEQUENCE, Length = 64
 - SEQUENCE, Length = 57
 - SET, Length = 11
 - SEQUENCE, Length = 9
 - OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 6 } countryName
 - PrintableString, Length = 2, Value = "DE"
 - SET, Length = 42
 - SEQUENCE, Length = 40
 - OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 10 } organizationName
 - PrintableString, Length = 33, Value = "ITSG TrustCenter fuer Arbeitgeber" ← Issuer
 - INTEGER, Length = 3, Value = 42384 (0xA590) ← Seriennummer des Signierer (Absender)
 - SEQUENCE, Length = 9
 - OBJECT IDENTIFIER, Length = 5, Value = { 1 3 14 3 2 26 } sha1 ← digestAlgorithm Kapitel 7.2.6.3
 - NULL
- (hier steht das optionale Datenfeld signedAttrs, siehe unten ...)
- SEQUENCE, Length = 13
 - OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840113549 1 1 1 } rsaEncryption ← signatureAlgorithm Kapitel 7.2.6.5
 - NULL
- + OCTET STRING, Length = 256 ← Datenfeld signature Kapitel 7.2.6.6

7.3.3 Teil 2a, signedAttrs

Nur der Vollständigkeit halber zeigen wir die Sequence des Datenfeldes `signerInfos`, das zwar optional, aber für die Auflösung einer Singnatur wichtig ist (nur hier wird der Zeitpunkt der Signatur ersichtlich). Dieses Datenfeld wird, wenn vorhanden, an die im oben gezeigten Beispiel an die bezeichnete Stelle eingefügt.

```
- SEQUENCE
- OBJECT IDENTIFIER contentType (1 2 840 113549 1 9 3)
- SET
  - OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
SEQUENCE
  OBJECT IDENTIFIER signingTime (1 2 840 113549 1 9 5)
  SET
    UTCTime '080520161940Z' ← Zeitpunkt der Signatur
SEQUENCE
  OBJECT IDENTIFIER messageDigest (1 2 840
  SET
    OCTET STRING
      CA 2D 5F 6A 9A B4 DD 17 3F 3E 93 BE 3F B9 13 C0 23 E2 CF 07
```

signedAttrs
Kapitel 7.2.6.4

Achtung: OPTIONAL!

7.4 Aufbau der Teildatenstruktur Content vom ContentType „EnvelopedData“

Der Typ `EnvelopedData` hat folgende Syntax:

```
EnvelopedData ::= SEQUENCE {  
    version CMSVersion,  
    originatorInfo [0] IMPLICIT OriginatorInfo OPTIONAL,  
    recipientInfos RecipientInfos,  
    encryptedContentInfo EncryptedContentInfo,  
    unprotectedAttrs [1] IMPLICIT UnprotectedAttributes OPTIONAL }  
  
RecipientInfos ::= SET OF RecipientInfo
```

7.4.1 Datenfeld „version“

Referenz: [IMTTP3] Tabelle 6#1,
[RFC 2630] Abschnitt 6.1

Das Feld `version` enthält die Versionsnummer der Syntax. Die Versionsnummer für die in der vorliegenden Spezifikation verwendete Syntax ist „0“. Die Versionsnummer richtet sich danach, ob das Feld `originatorInfo` vorhanden ist, nach der Versionsnummer für die Datenstruktur `RecipientInfos` und ob das Feld `unprotectedAttrs` vorhanden ist.

Profilierung:

Gemäß [IMTTP3] Tabelle 3.3#1 soll das Datenfeld „version“ den Wert „0“ erhalten, was impliziert, dass die Datenfelder `originatorInfo` und `unprotectedAttrs` entfallen müssen und dass alle Strukturen vom Typ `RecipientInfos` die Version „0“ haben.

7.4.2 Datenfeld „originatorInfo“

Referenz: [IMTTP3] Tabelle 6

Das Feld `originatorInfo` enthält üblicherweise Informationen über den Aussteller der Nachricht. Es ist optional, da es vom verwendeten Algorithmus für das Schlüsselmanagement abhängig ist, ob dieses Feld verwendet werden muss. Falls nur Algorithmen verwendet werden, deren Unterstützung gefordert oder empfohlen wird, wird dieses Feld nicht benötigt.

Profilierung:

Von diesem optionalen Feld soll in den Nachrichten kein Gebrauch gemacht werden. Ohne Angabe einer `originatorInfo` muss es aus der Datenstruktur `EnvelopedData` entfallen; vgl. auch 7.4.1.

7.4.3 Datenfeld „RecipientInfos“

Referenz: [IMTTP3] Tabellen 6 und 7
[RFC 2630], Abschnitt 6.2

Die Datenstruktur `RecipientInfos` enthält Informationen für einen oder mehrere Empfänger der Nachricht. Für das Feld `RecipientInfos` wird folgende Syntax verwendet:

Vgl. [IMTTP3] Tabelle 7 und [RFC 2630], Abschnitt 6.2:

```
RecipientInfos ::= SET OF RecipientInfo

RecipientInfo ::= CHOICE {
    ktri KeyTransRecipientInfo,
    kari [1] KeyAgreeRecipientInfo,
    kekri [2] KEKRecipientInfo }
```

Wie in [IMTTP3] wird ausschließlich die Datenstruktur „KeyTransRecipientInfo“ verwendet.
Vgl. [IMTTP3] Tabelle 8:

```
KeyTransRecipientInfo ::= SEQUENCE {
    version CMSVersion,
    rid RecipientIdentifier,
    keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
    encryptedKey EncryptedKey }

RecipientIdentifier ::= CHOICE {
    issuerAndSerialNumber IssuerAndSerialNumber,
    subjectKeyIdentifier [0] SubjectKeyIdentifier }

EncryptedKey ::= OCTET STRING
```

Für jeden Empfänger wird eine Datenstruktur vom Typ `KeyTransRecipientInfo` verwendet; alle anderen Schlüsseltransport-Alternativen werden von [IMTTP3] nicht berücksichtigt. Die anderen Datenstrukturen der Auswahl beziehen sich auf ein Schlüsselmanagement mit Diffie-Hellman-Schlüsseln bzw. symmetrischen Schlüsseln. Für dieses Profil ist nur das RSA-Schlüsselaustauschverfahren vorgesehen.

Die Datenstruktur könnte auch für den Absender verwendet werden, falls er selbst in der Lage sein will, die Nachricht wieder zu entschlüsseln.

7.4.3.1 Teildatenfeld „version“

Referenz: [IMTTP3] Tabelle 8.#1
[RFC 2630], Abschnitt 6.2

Profilierung:

Der Wert für das Feld `version` der Datenstruktur `KeyTransRecipientInfo`, der die Versionsnummer der CMS-Syntax angibt, muss stets „0“ sein; vgl. [IMTTP3] Tabelle 8, Notes [1]. Entsprechend Kapitel 6.2.1 [RFC 2630] ist dies der Fall, wenn die Variante `issuerAndSerialNumber` gewählt wird. (s. u.)

7.4.3.2 Datenfeld „rid“

Referenz: [IMTTP3] Tabelle 8.#2
[RFC 2630], Abschnitt 6.2

Mit dem Datenfeld das Feld `rid` wird das Zertifikat des Empfängers (und damit der öffentliche Schlüssel des Empfängers) identifiziert.

Profilierung:

Für das Datenfeld `rid` darf nur die Variante `issuerAndSerialNumber` gewählt werden; vgl. [IMTTP3] Tabelle 8, Notes [2]. Das Zertifikat wird damit durch den Namen der ausstellenden CA und die Seriennummer eindeutig bestimmt. Dies stimmt mit den Vorgaben zur Identifizierung der Signaturschlüssel von CAs überein.

7.4.3.3 Teildatenfeld „keyEncryptionAlgorithm“

Referenz: [IMTTP3] Tabelle 8.#3
[RFC 2630], Abschnitt 6.2

Das Feld `keyEncryptionAlgorithm` enthält den OID für den Verschlüsselungsalgorithmus, mit dem der Nachrichtenschlüssel verschlüsselt wird.

Profilierung:

Als Verschlüsselungsalgorithmus für den Nachrichtenschlüssel ist von der [SecSchnitt] RSA vorgesehen; vgl. Abschnitt 8.4.

7.4.3.4 Teildatenfeld „encryptedKey“

Referenz: [IMTTP3] Tabelle 8.#1
[RFC 2630], Abschnitt 6.2

Das Feld `encryptedKey` enthält den verschlüsselten Nachrichtenschlüssel. Dies ist das Ergebnis der Verschlüsselung des Nachrichtenschlüssels (content-encryption key) mit dem öffentlichen Schlüssel des Empfängers.

7.4.4 Datenfeld „ encryptedContentInfo“

Referenz: [IMTTP3] Tabellen 6.#4 und 9
[RFC 2630], Abschnitt 6.1

Das Feld `encryptedContentInfo` der Datenstruktur `EnvelopedData` enthält die verschlüsselten Daten. Es wird folgende Datenstruktur verwendet:

```
EncryptedContentInfo ::= SEQUENCE {
    contentType ContentType,
    contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,
    encryptedContent [0] IMPLICIT EncryptedContent OPTIONAL }

EncryptedContent ::= OCTET STRING
```

Die folgenden Unterabschnitte beschreiben die einzelnen Teildatenfelder der Datenstruktur `EncryptedContentInfo`.

7.4.4.1 Teildatenfeld „contentType“

Referenz: [IMTTP3] Tabelle 9#1
[RFC 2630], Abschnitt 6.1

Das Feld `contentType` enthält den Inhaltstyp der verschlüsselten Daten.

Profilierung:

Gemäß [IMMTP3] wird hierfür genau ein `contentType` verwendet: „id-data“, mit der OID 1.2.840.113549.1.7.1, die angibt, dass nicht interpretierte binäre Daten verschlüsselt und in das Teildatenfeld `encryptedContent` eingetragen wurden.

7.4.4.2 Teildatenfeld „contentEncryptionAlgorithm“

Referenz: [IMTTP3] Tabelle 9#2
[RFC 2630], Abschnitt 6.1; sowie [RFC 3565], Abschnitte 3 und 4

Das Feld `contentEncryptionAlgorithm` identifiziert den Verschlüsselungsalgorithmus, mit dem die Daten, die in das Teildatenfeld `encryptedContent` eingetragen werden, verschlüsselt werden.

Profilierung:

Als Verschlüsselungsalgorithmen für die Daten sind Triple-DES (`des3-cbc`, auslaufend) sowie AES (`aes256-CBC`) vorgesehen; vgl. Abschnitt 8.3 "Verschlüsselungsalgorithmen für Daten (Content Encryption Algorithms)". Dabei ist die Migrationsstrategie gemäß [SecSchnitt] zu beachten.

7.4.4.3 Teildatenfeld „encryptedcontent“

Referenz: [IMTTP3] Tabelle 9#3
[RFC 2630], Abschnitt 6.1

Das Feld `encryptedContent` enthält die verschlüsselten Daten, also das Ergebnis der Verschlüsselung mit dem Nachrichtenschlüssel.

7.4.5 Datenfeld „unprotectedAttrs“ (entfällt!)

Referenz: [IMTTP3] Tabelle 6#5

Das Feld `unprotectedAttrs` kann Attribute enthalten, die nicht verschlüsselt werden.

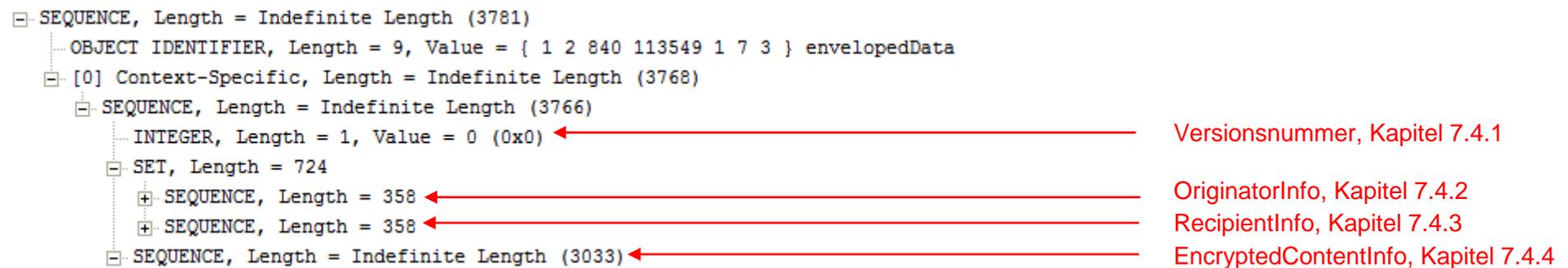
Profilierung:

Von diesem Feld darf in den Nachrichten kein Gebrauch gemacht werden, es muss aus der Datenstruktur `EnvelopedData` entfallen; vgl. auch 7.4.1.

7.5 Erklärung einer PKCS#7-verschlüsselten Nachricht gemäß Kapitel 7.4

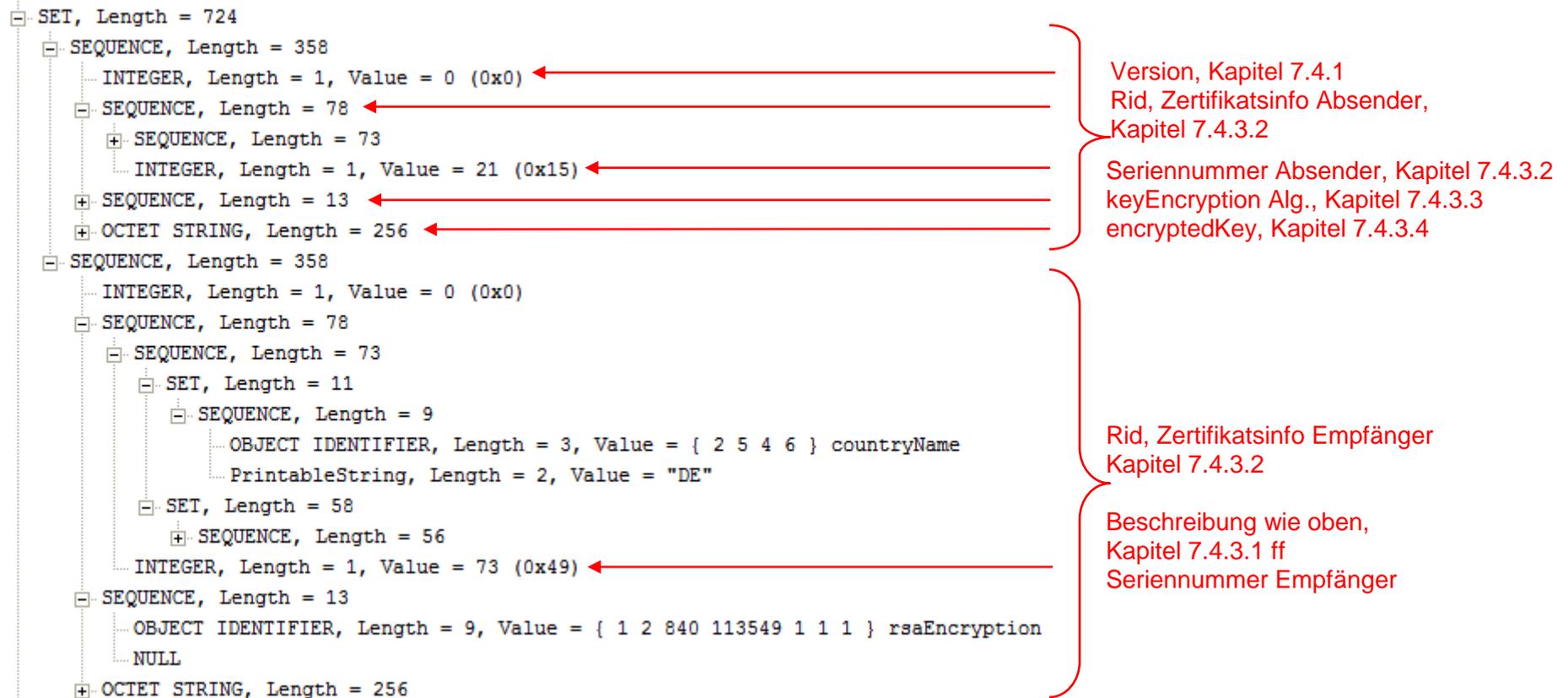
Das hier gezeigte Beispiel einer PKCS#7-verschlüsselten Nachricht mit der Hash-Variante SHA-1 und Triple-DES-Verschlüsselung ist zunächst auf die drei wesentlichen Sequenzen zusammengezogen und entsprechend kommentiert. In den weiteren Teilen werden die wesentlichen Details genauer spezifiziert.

7.5.1 Teil 1, envelopedData



7.5.2 Teil 2, OriginatorInfo und RecipientInfo

Hier werden die wesentlichen Details in den beiden Sequenzen für Absender und Empfänger gelistet.



7.5.3 Teil 3, EncryptedContentInfo

```
[-] SEQUENCE, Length = Indefinite Length (3033)
  [-] OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 7 1 } data ← ContentType, Kapitel 7.4.4.1
  [-] SEQUENCE, Length = 20
    [-] OBJECT IDENTIFIER, Length = 8, Value = { 1 2 840 113549 3 7 } des-EDE3-CBC ← ContentEncryptionAlgorithm, Kapitel 7.4.4.2
    [-] OCTET STRING, Length = 8
      1: 67 C1 87 3E 05 E7 69 C4      gA.>.giD
    [-] [0] Context-Specific, Length = Indefinite Length (2998) ← Encryptedcontent, Kapitel 7.4.4.3
      [+] OCTET STRING, Length = 1000
      [+] OCTET STRING, Length = 1000
      [+] OCTET STRING, Length = 976
```

7.6 Transport der der PKCS#7-verschlüsselten Nachricht

Nachdem die signierte und verschlüsselte PKCS#7-Nachricht die Kombination eines Datenobjekt vom Typ `SignedData` und eines vom Typ `EnvelopedData` (vgl. 7.1 „Aufbau der PKCS#7-signierten und verschlüsselten Nachricht“) zusammengestellt wurde, liegt ein solches Datenobjekt als ASN.1-Struktur vor. Es folgt eine DER-Kodierung, die als Transportsicherung dient. Der daraus resultierende Bytestream wird in eine Datei abgelegt, die folgende Dateiendung aufweist:

Nachrichtenart	Dateiendung
Verschlüsselte Nachricht	<keine> ¹²

Das physikalisch so gesicherte PKCS#7-Datenobjekt kann auf verschiedenen Transportwegen gesendet werden.

7.6.1 Transportformat

Referenz: [IMTTP3], Abschnitt 2.1.1,
[SecSchnitt] Abschnitt 5.3

Eine verschlüsselte Nachricht als PKCS#7-Datenobjekt wird für den Datenaustausch zwischen zwei Kommunikationspartnern eingesetzt, um eine Nutzdatendatei verschlüsselt auszutauschen. Eine verschlüsselte Nachricht als PKCS#7-Datenobjekt wird in einer Datei abgelegt, die keine Dateiendung aufweist. Physikalisch handelt es sich um eine Binärdatei, die auf den weiteren Transportwegen als eben solche zu behandeln ist.

7.6.2 Transportwege

Referenz: [SecSchnitt] Abschnitt 5.3

Für die verschlüsselten Nachrichten wird ausschließlich die Übertragung der Nutzdaten per E-Mail betrachtet.

Transportweg	Details
E-Mail	Datei ohne Dateiendung mit der PKCS#7-Datenstruktur zur Übertragung der Nutzdaten; sie wird als Anhang an eine Mail versendet.

¹² Die Dateinamen der verschlüsselten Nachrichten im Gesundheitswesen werden von den Spitzenverbänden der GKV als Identifikatoren für Verfahrensspezifische Ausprägungen genutzt. Diese werden als Verfahrenskennungen bezeichnet. Eine Dateiendung ist für den verschlüsselten Nachrichtenaustausch nicht vorgesehen. Alle verschlüsselten Nutzdaten werden ohne ein Suffix versendet. Die verschlüsselten Nutzdaten haben im Deutschen Gesundheitswesen kein Suffix; vgl. „Richtlinien für den Datenaustausch“, siehe <http://www.datenaustausch.de>.

8 Krypto-Algorithmen

Alle für ISIS-MTT verwendbaren Krypto-Algorithmen werden in [IMTTP6] spezifiziert.

Im vorliegenden Dokument werden die Krypto-Algorithmen beschrieben, die gemäß [SecSchnitt] anzuwenden sind.

Krypto-Algorithmen lassen sich nach folgenden Gruppen (vgl. [IMTTP6]) voneinander unterscheiden, von denen nicht alle Gruppen für dieses Profil relevant sind:

- Einweg-Hash-Funktionen (One-Way Hash Functions),
- Signaturalgorithmen (Signature Algorithms),
- Verschlüsselungsalgorithmen für Daten (Content Encryption Algorithms),
- Verschlüsselungsalgorithmen für den Nachrichtenschlüssel (Key Encryption Algorithms),
- Verschlüsselungsalgorithmen des öffentlichen Schlüssels (Subject Key Public Algorithms) und
- Authentisierungsalgorithmen für PKI-Nachrichten (Message Authentication Algorithms) – nicht relevant.

8.1 Einweg-Hash-Funktionen (One-Way Hash Functions)

Referenz: [SecSchnitt], Abschnitt 2.1.3
[IMTTP6], Tabelle 1
[FIPS180-3]

[SecSchnitt], Abschnitt 2.1.3 sieht als Hashfunktionen die in [FIPS180-3] definierten Algorithmen SHA-1 und SHA-256 vor.

Name	OID
SHA-1	1.3.14.3.2.26
SHA-256	2.16.840.1.101.3.4.2.1

Die Hashfunktionen werden dazu verwendet, aus einer gegebenen Nachricht einen sogenannten Message Digest zu errechnen. Ein Message Digest ist ein Wert, der weitgehend eindeutig einer Nachricht zugeordnet werden kann (Kollisionsresistenz der Hashfunktion). Aus dem Message Digest lässt sich jedoch die zugrunde liegende Nachricht nicht wieder zurückberechnen (Einweg-Funktion).

- Eingabe: Die zu signierende Nachricht.
- Ausgabe: Eine Zeichenfolge fester Bitlänge.

Diese Funktion wird nicht explizit in Zertifikaten oder Nachrichten angegeben. Sie wird jedoch als Hashfunktion bei der Signaturerstellung verwendet (s. Abschnitt 8.2). Bei der Realisierung ist darauf zu achten, dass der durch den oben angegebenen Objektbezeichner referenzierte Algorithmus verwendet wird.

Profilierung:

Im vorliegenden Verfahren werden Hashwerte als so genannte Fingerprints verwendet. Sie dienen dazu die Einmaligkeit eines Schlüssels im Verfahren und die Zugehörigkeit eines Schlüssels zum jeweiligen Antragsteller schneller überprüfen zu können. Hierzu wird der Hashwert über den zur Zertifizierung vorgelegten öffentlichen Schlüssel gebildet und als Ausdruck dem schriftlichen Antrag beigelegt.

Der öffentliche Schlüssel ist sowohl in einem X.509-Zertifikat als auch in einem PKCS#10-Request als `BITSTRING` mit der Bezeichnung `subjectPublicKey` in der Struktur `subjectPublicKeyInfo` enthalten. Der Fingerprint ist ausschließlich über den kompletten `BITSTRING` zu bilden.

8.2 Signaturalgorithmen (Signature Algorithms)

Referenz: [SecSchnitt], Abschnitte 2.1.2, 2.1.3 und 2.1.5
[IMTTP6], Tabelle 2

[SecSchnitt], Abschnitt 2.1.3 sieht als Signaturalgorithmus RSA mit SHA-1 bzw. SHA-256 vor.

Name	OID
sha-1WithRSAEncryption	1.2.840.113549.1.1.5
sha256withRSAEncryption	1.2.840.113549.1.1.11

Als öffentlicher Exponent des RSA-Schlüssels wird die 4. Fermatsche Zahl (0x10001 bzw. 65537 dezimal) verwendet.

Der Signaturalgorithmus wird zum Signieren von Daten verwendet. Er ist eine Kombination einer Hashfunktion (siehe 8.1) und einem Public-Key-Algorithmus (siehe 8.5).

- Eingabe: Die zu signierende Nachricht und der private Schlüssel des Signierenden.
- Ausgabe: Die digitale Signatur.

Der zugehörige Objektbezeichner wird im Feld `Algorithm` der Struktur `Signature` eines Zertifikates oder PKCS#10-Requests eingetragen.

8.3 Verschlüsselungsalgorithmen für Daten (Content Encryption Algorithms)

Referenz: [SecSchnitt], Abschnitt 2.1.1
[IMTTP6x], Tabelle 3

[SecSchnitt], Abschnitt 2.1.1 sieht Triple-DES (vgl. [IMTTP6x], Tabelle 3.#2: des-ede3-cbc) sowie AES (vgl. [RFC 3565]) mit 256 Bit Schlüssellänge und CBC-Betriebsmodus vor.

Name	OID
des-ede3-cbc	1.2.840.113549.3.7
id-aes256-CBC	2.16.840.1.101.3.4.1.42

Der Verschlüsselungsalgorithmus ist der eigentliche Algorithmus mit dem Daten ver- und entschlüsselt werden. Hier wird ein symmetrischer Algorithmus verwendet, bei dem Verschlüsseler und Entschlüsseler über den gleichen Schlüssel verfügen müssen.

	Verschlüsselung	Entschlüsselung
• Eingabe:	Die zu verschlüsselnden Daten	Die verschlüsselten Daten
	Der Nachrichtenschlüssel	Der Nachrichtenschlüssel
• Ausgabe:	Die verschlüsselten Daten	Die entschlüsselten Daten

Der Nachrichtenschlüssel ist der Schlüssel, mit dem die Daten verschlüsselt werden. Für den Triple-DES-Algorithmus gilt eine Schlüssellänge von 168 Bit; für AES gilt eine Schlüssellänge von 256 Bit.

Der zugehörige Objektbezeichner wird im Feld `contentEncryptionAlgorithm` der Struktur `EncryptedContentInfo` in der PKCS#7-Nachricht eingetragen.

8.4 Verschlüsselungsalgorithmen für den Nachrichtenschlüssel (Key Encryption Algorithms)

Referenz: [SecSchnitt], Abschnitte 2.1.2, 2.1.5
[IMTTP6], Tabelle 5
[PKCS#1]

Zur Verschlüsselung des Nachrichtenschlüssels wird RSA eingesetzt; vgl. [SecSchnitt], Abschnitt 2.1.2 und [IMTTP6], Tabelle 4.#1: `rsaEncryption`.

Name	OID
<code>rsaEncryption</code>	1.2.840.113549.1.1.1

Als öffentlicher Exponent des RSA-Schlüssels wird die 4. Fermatsche Zahl (0x10001 bzw. 65537 dezimal) verwendet.

Der Austausch des Nachrichtenschlüssels erfolgt mit Hilfe eines asymmetrischen Verschlüsselungsverfahrens.

	Verschlüsselung	Entschlüsselung
• Eingabe:	Der Nachrichtenschlüssel	Der verschlüsselte Nachrichtenschlüssel
	Der öffentliche Schlüssel des Empfängers	Der private Schlüssel des Empfängers
• Ausgabe:	Der verschlüsselte Nachrichtenschlüssel	Der Nachrichtenschlüssel

Der zugehörige Objektbezeichner wird im Feld `Algorithm` der Struktur `subjectPublicKeyInfo` eines Zertifikates oder PKCS#10-Requests eingetragen.

8.5 Algorithmen zur Nutzung des öffentlichen Schlüssels (Subject Public Key Algorithms)

Referenz: [SecSchnitt], Abschnitte 2.1.2, 2.1.4 und 2.1.5
[IMTTP6], Tabelle 5
[PKCS#1]

[SecSchnitt], Abschnitte 2.1.2 und 2.1.4 sieht RSA vor (vgl. [IMTTP6], Tabelle 5.#1: `rsaEncryption`).

Name	OID
RsaEncryption	1.2.840.113549.1.1.1

Als Schlüssellänge gibt [SecSchnitt], Abschnitt 2.1.4 für Zertifizierungsstellen (PCA und CA) und für Zertifikatsinhaber 2048 bit vor.

Als öffentlicher Exponent des RSA-Schlüssels wird die 4. Fermatsche Zahl (0x10001 bzw. 65537 dezimal) verwendet.

Je nach Anwendungsumfeld (Signatur oder Ver-/Entschlüsselung) wird dieser Algorithmus wie in 8.2 oder 8.4 verwendet.

Der zugehörige Objektbezeichner wird im Feld `Algorithm` der Struktur `subjectPublicKeyInfo` eines Zertifikates oder PKCS#10-Requests eingetragen.

Profilierung:

Gemäß der [SecSchnitt] Abschnitt 2.1.4 werden die Schlüssellängen für Teilnehmer ebenfalls mit 2048 Bit genutzt. Das BSI erstellt jedes Jahr ein Papier über die Eignung von Kryptoalgorithmen die den Vorgaben des deutschen Signaturgesetzes genügen.

9 LDAP

In diesem Abschnitt werden die LDAP-Verzeichnisdienste beschrieben, die gemäß [SecSchnitt2], Abschnitt 2.3 zur Anwendung kommen. Der Verzeichnisdienst basiert auf der aktuellen Version LDAP v3, die im [RFC 2251] dokumentiert ist und berücksichtigt die Spezifikationen von [IMTTP4]. Weiterhin werden sichere LDAP-Verbindungen über LDAPS (auch als LDAP-SSL bezeichnet) konform zu [RFC 2830] angeboten.

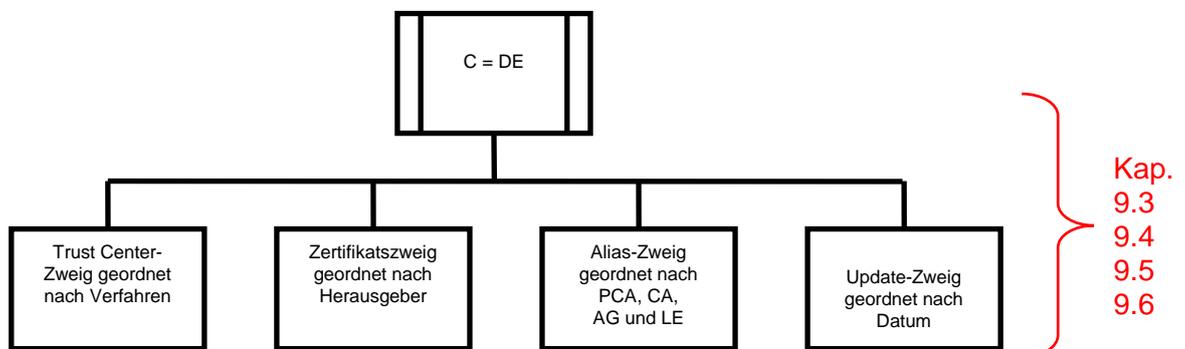
9.1 Aufbau des LDAP Servers

Referenz: [SecSchnitt], Abschnitt 2.3
[RFC 2251]
[RFC 2830]
[IMTTP4]

Das LDAP-Verzeichnis ist in einem DIT (Directory Information Tree) in mehrere Zweige aufgeteilt:

Der Trust Center-Zweig ist für eine manuelle Suche aus Sicht eines Trust Centers optimiert. Der Zertifikatszweig und der Alias-Zweig sind für eine maschinelle Suche optimiert. Der Alias-Zweig unterstützt bei der Suche ohne Kenntnis der zugeordneten CA mittels eines Alias-Eintrags. Der Update-Zweig unterstützt das Trust Center bei der Verwaltung von Aktualisierungen.

Die Hierarchie ist baumförmig aufgebaut. Folgendes Schaubild veranschaulicht dies:



Grobstruktur des LDAP Servers mit Toplevel des DIT

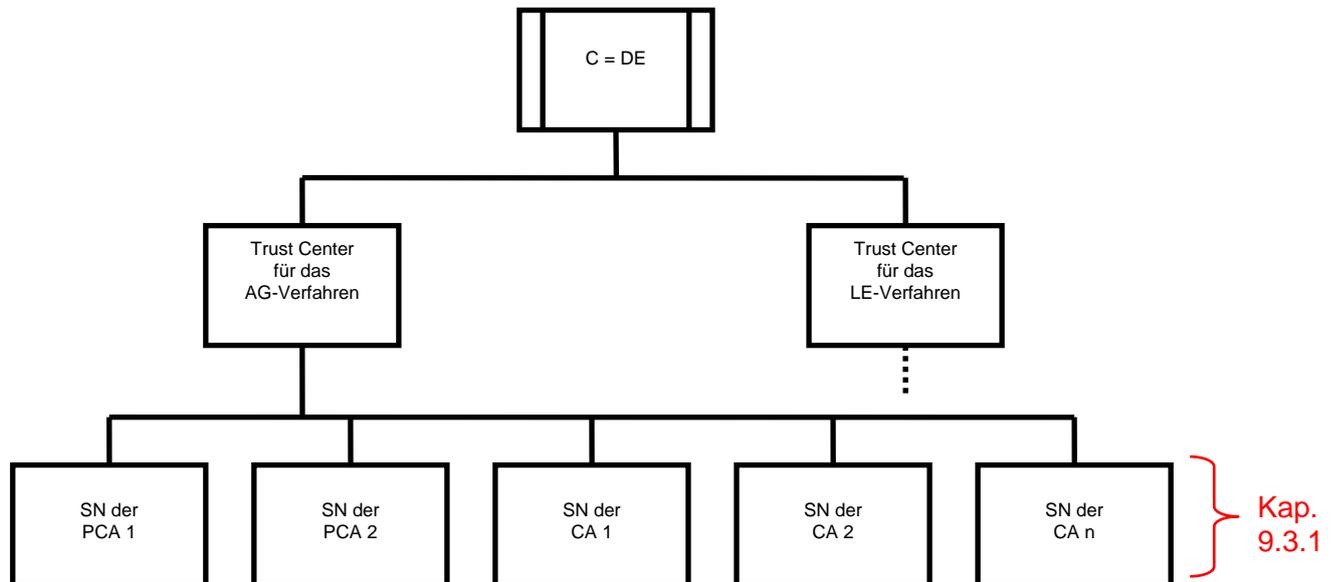
Der LDAP-Server wird über die Standard-Ports 389 (LDAP) und 636 (LDAPS) betrieben.

9.2 Aufbau der Wurzel

Name	Object Class	Attribute
De	Country	• countryName=de

9.3 Aufbau des Trust Center Zweigs (geordnet nach Datenaustauschverfahren)

Der Trust Center Zweig ist nach den Datenaustauschverfahren für Arbeitgeber und Leistungserbringer aufgeteilt. Die nachfolgende Abbildung zeigt den Directory Information Tree mit den beiden Datenaustausch-verfahren und den zugeordneten PCA's und CA's.



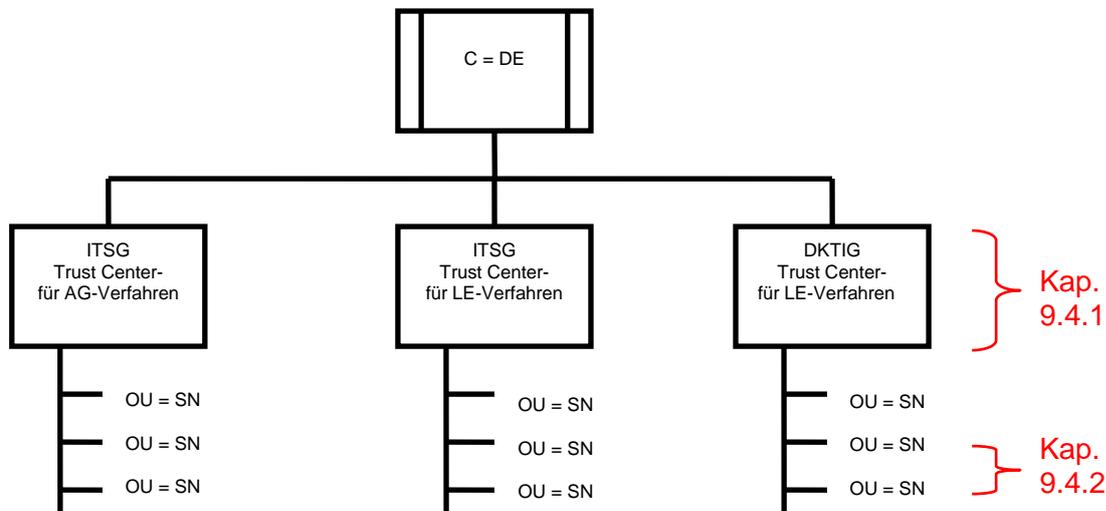
9.3.1 Aufbau der PCA- und CA-Ebene

Für jede PCA und CA wird ein eigenes Objekt der Klasse „Organization“ angelegt.

Name	Object Class	Attribute
<Seriennummer aus dem Zertifikat>	Organization	<ul style="list-style-type: none"> • Trust Center-Name • Seriennummer des Zertifikats • Zertifikat binär

9.4 Aufbau des Zertifikatzweigs (geordnet nach Herausgeber)

Die nachfolgende Abbildung zeigt den Directory Information Tree für den Zertifikatzweig. Auf der Zertifikat-Issuer Ebene befinden sich alle CA's. Eine Ebene tiefer sind dann alle dem Herausgeber (Issuer) zugeordneten Zertifikate angeordnet.



9.4.1 Aufbau der Issuer-Ebene

Für jede PCA und CA wird ein eigenes Objekt der Klasse „Organization“ angelegt. Auf der Issuer-Ebene wird in den einzelnen CA-Zweigen die zugeordnete Sperrliste (certificateRevocationList) als Binärdatei bereitgestellt.

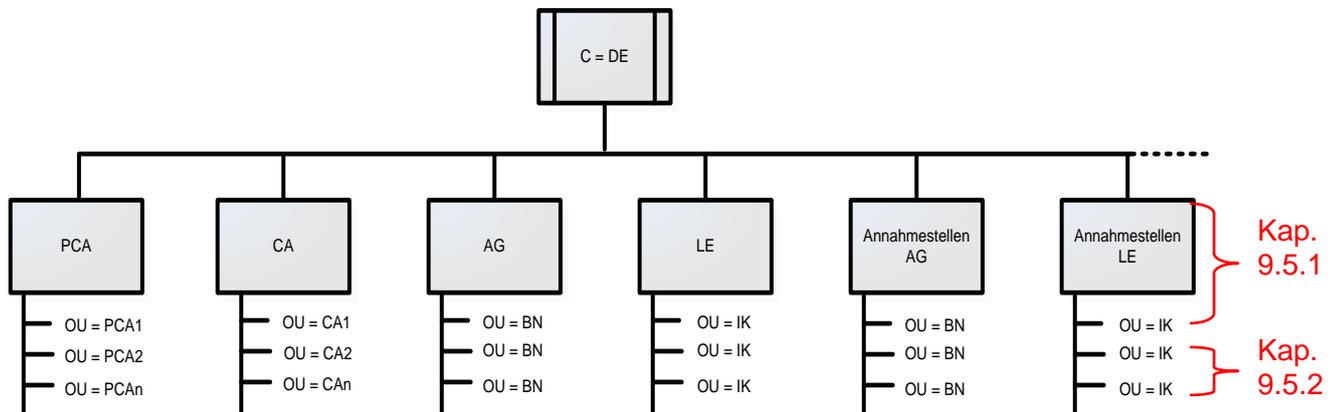
Name	Object Class	Attribute
<OrganizationName aus dem Zertifikat>	Organization Unit	<ul style="list-style-type: none"> certificateRevocationList = <CRL der CA> Seriennummer des Zertifikats

9.4.2 Aufbau der Zertifikats-Ebene

Name	Object Class	Attribute
<Seriennummer des Zertifikats>	Organization Unit	<ul style="list-style-type: none"> Firmenname Institutionskennzeichen oder Betriebs- bzw. Zahlstellennummer Seriennummer des Zertifikats Zertifikat (binär)

9.5 Aufbau des Alias-Zweigs

Die nachfolgende Abbildung zeigt den Directory Information Tree für den suchoptimierten Alias-Zweig geordnet nach PCA, CA, AG und LE sowie Annahmestellen AG/LE. Die Alias-Zweige beinhalten nicht die Binärdateien, stellen aber alle wesentlichen Attribute eines Zertifikats für eine Recherche bereit. Eine Referenz verweist auf die Binärdateien der Zertifikate in dem zugeordneten Issuer-Zweig. Damit wird eine Redundanz der Daten in der LDAP-Struktur verhindert.



9.5.1 Aufbau der PCA-, CA-, AG-, LE- und Annahme-Ebene

Für die PCA, CA, AG und LE sowie jede Annahmestelle AG und LE wird ein eigenes Objekt der Klasse „Organization“ angelegt.

Name	Object Class	Attribute
<OrganizationName aus dem Zertifikat>	Organization	<ul style="list-style-type: none"> Trust Center-Name oder Institutionskennzeichen oder Betriebs- bzw. Zahlstellenummer

9.5.2 Aufbau der Verweis-Ebene

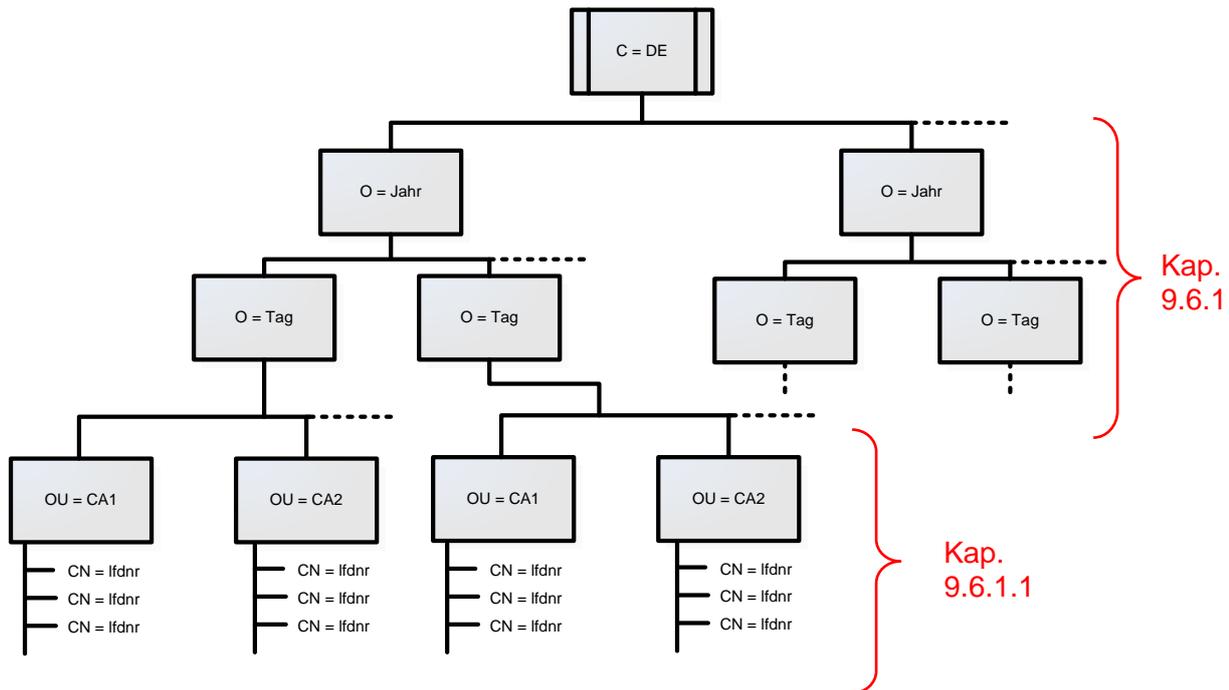
Für jedes Zertifikat, das von einem Trust Center zu einem Institutionskennzeichen oder einer Betriebs- bzw. Zahlstellenummer erstellt wurde, wird ein eigenes Objekt der Klasse „Organization Unit“ angelegt. Der Objektname ist die Seriennummer des jeweiligen Zertifikats.

Name	Object Class	Attribute
<Seriennummer des Zertifikats>	Organization Unit	<ul style="list-style-type: none"> Trust Center-Name oder Firmenname Seriennummer des Zertifikats Referenz zum Zertifikat (seeAlso) Gültigkeit des Zertifikats „von“ Gültigkeit des Zertifikats „bis“ Verschlüsselungsverfahren Ansprechpartner im Fall AG oder LE

9.6 Aufbau des Update-Zweig

Die nachfolgende Abbildung zeigt den Directory Information Tree für den Update-Zweig. Die Updates werden zur besseren Übersicht und Sortierung in Zweige nach dem Jahr und in Unterzweigen für jeden Tag unterhalb des zugehörigen Trust Centers (CA) fortlaufend angelegt.

HINWEIS: Die Update-Zweige nehmen in der überarbeiteten LDAP-Struktur nur noch eine Verwaltungsrolle für das Trust Center ein und erheben keinen Anspruch auf alle Änderungen zum letzten Stand, denn es werden immer nur neue Zertifikate hinzugefügt, ein Löschen erfolgt hier nicht.



9.6.1 Aufbau der Datums-Ebene

Für jede inkrementelle Speicherung neuer Zertifikate wird ein Objekt der Klasse „Organization“ angelegt. Der Organization Name wird aus dem aktuellen Datum generiert und als String in der Form YYYY für das Jahr und YYYYMMDD für einen Tag (e.g. “20061117“) dargestellt.

Name	Object Class	Attribute
<Organization Name aus dem aktuellen Datum>	Organization	<ul style="list-style-type: none"> Trust Center-Name

9.6.1.1 Aufbau der CA- und Zertifikats-Ebene

Der Aufbau der CA- und Zertifikats-Ebene entspricht dem Zertifikatszweig, Details sind unter dem Punkt 10.4 zu entnehmen.

9.7 Zugriffsrechte und -rollen

Bei den Zugriffen wird zwischen zwei Zugriffsarten unterschieden:

- Anonymer Zugriff
- Zugriff mit Anmeldung und Benutzerrechten

9.7.1 Anonyme Zugriffsrechte

Der anonyme Zugriff ist vor allem für die Arbeitgeber und Leistungserbringer gedacht, die über diesen Zugriff die neuesten Zertifikate der Annahmestellen vom LDAP-Server herunterladen und diese gegen die Sperrlisten auf Gültigkeit prüfen können.

Für diese LDAP-Verbindung wird keine Authentisierung benötigt, sie erfolgt über den „Anonymous Account“.

Der Anonymous Account hat ausschließlich Lese-Zugriff auf die Alias-Ebene mit den Annahmestellen in allen Teilästen. Kein Zugriff besteht auf die Einträge anderer Arbeitgeber oder Leistungserbringer. Schreibzugriffe sind ebenfalls nicht möglich.

Die Zugriffe sind sowohl über LDAP (Port 389), als auch über LDAPS (Port 636) möglich. Diese Anmeldeart ist derzeit noch nicht frei geschaltet.

9.7.2 Anmeldung mit Zugriffsrechten (Annahmestellen der GKV)

Die Annahmestellen der GKVen erhalten vom LDAP-Betreiber eine Benutzerkennung mit Kennwort, mit der sie sich auf dem LDAP-Server anmelden können. Mit dieser Kennung erhält jede Annahmestelle Lesezugriff auf den kompletten LDAP-Server. Schreibzugriffe sind nicht möglich.

Die Benutzer werden in einer separaten Datenbank gepflegt.

Es wird für diese authentisierten Verbindungen aus Sicherheitsgründen empfohlen, LDAPS (Port 636) zu verwenden. Alternativ kann über LDAP (Port 389) zugegriffen werden.

9.8 Suchfunktionen

Die Suche nach Binärdateien der Teilnehmer-Zertifikate ist über das Institutionskennzeichen oder über die Betriebs- bzw. Zahlstellenummer sowie über die Seriennummer in den Zertifikats-Zweigen möglich. Dabei ist wichtig, dass der Präfix „BN“ bzw. „IK“ zur Nummer mit angegeben wird. Auch die Binärdateien der Sperrlisten (CRL) können in den Zertifikats-Zweigen gesucht werden. Für die Recherche nach bestimmten Attributen ist zudem eine Suche nach dem Firmennamen, Ansprechpartner, Gültigkeit und Verschlüsselungsverfahren in den Alias-Zweigen möglich.

HINWEIS: Gesucht werden kann über Standard LDAP-Routinen in allen Teilästen, in denen der Benutzer Leserechte hat.

9.9 Zertifikatshandling

9.9.1 Sperrung eines Zertifikats

Sobald ein Zertifikat gesperrt wird, wird die aktualisierte Sperrliste bei der zugehörigen CA veröffentlicht und der zu dem Zertifikat zugehörige Benutzereintrag im LDAP-Server gelöscht.

Wird ein CA-Zertifikat gesperrt, wird der komplette Zweig der CA und alle Benutzer-Zertifikate, die von dieser gesperrten CA erzeugt wurden, vom LDAP-Server gelöscht.

Wird ein PCA-Zertifikat gesperrt, wird der komplette Zweig der PCA, alle von dieser PCA erzeugten CAs mit allen Teilnehmern der entsprechenden CA vom LDAP-Server gelöscht.

Somit ist die Prüfung, ob ein Zertifikat gesperrt wurde, sowohl über die zugehörige Sperrliste, als auch über einen Positiv-Abgleich mit dem LDAP-Server möglich.

9.9.2 Abgelaufene Zertifikate

Abgelaufene Benutzerzertifikate werden vom LDAP-Server gelöscht. Bei abgelaufenen CA- oder PCA-Zertifikaten wird der komplette Teilast vom LDAP-Server gelöscht. Dieser Abgleich auf abgelaufene Benutzerzertifikate wird nicht über die Update-Zweige angewendet!

9.9.3 Abholung eines einzelnen Zertifikats vom LDAP-Server

Um Zertifikate und weitere Informationen vom LDAP-Server zu holen, kann über unterschiedliche Wege erfolgen. Prinzipiell müssen die Anfragen an den LDAP-Server nach Ver- oder Entschlüsseln unterschieden werden. Beim Verschlüsseln liegt in der Regel nur die Betriebs- bzw. Zahlstellenummer oder das Institutionskennzeichen des Empfängers vor, wogegen beim Entschlüsseln die Seriennummer und das Issuer-Trust Center bekannt ist (aus der verschlüsselten Nachricht). Aus diesen Anforderungen bieten sich die folgenden Suchwege an:

- Zur Verschlüsselung kann nach dem zugehörigen Institutionskennzeichen oder der zugehörigen Betriebs- bzw. Zahlstellenummer entweder im Zertifikatszweig (siehe Kapitel 9.4.2) oder im Aliaszweig AG oder LE (siehe Kapitel 9.5.2) gesucht werden. Als Suchergebnis werden 0-n Zertifikatsinformationen aufgelistet. Im Zertifikatszweig stehen die Binaries direkt zur Verfügung. Im Aliaszweig muss zunächst aus den gelieferten Informationen anhand der Attribute (von – bis, Verfahren, etc.) der gewünschte Verweis (= SeeAlso) selektiert werden und im zweiten Zugriffsschritt werden anhand der SeeAlso-Informationen die eindeutigen Zertifikatsinformationen (binary) im Zertifikatszweig gelesen.
- Beim Entschlüsselungsvorgang kann zum Verifizieren einer Signatur anhand der Seriennummer und dem Trust Center-Namen auf die eindeutigen Informationen des Zertifikates zugegriffen werden. Dabei wird direkt der Zertifikatszweig ausgelesen (siehe Kapitel 9.4.2).

Die CA- und PCA-Zertifikate können wahlweise über den Objektnamen oder das Attribut „serialnumber“ (Kapitel 9.3.1) gesucht werden. Dort liegt im Attribut „caCertificate“ das zugehörige Zertifikat.

Zertifikate, die vom LDAP-Server geladen werden, sind zu diesem Zeitpunkt immer gültig.

9.9.4 Abholung einer Sperrliste vom LDAP-Server

Um eine Sperrliste für Enduser-Zertifikate abholen zu können, muss zunächst der zugehörige CA-Zertifikatseintrag über den Objektnamen (Kapitel 9.4.1) gesucht werden. Dort ist im Attribut „CertificateRevocationList“ die zugehörige CRL zu finden.

Um die Gültigkeit eines CA-Zertifikats zu prüfen, muss im Eintrag des zugehörigen PCA-Zertifikats die „authorityRevocationList“ heruntergeladen werden.

9.9.4.1 Möglichkeiten zur Replizierung der LDAP-Daten

Für Annahmestellen, die eine eigene Kopie des LDAP-Servers betreiben möchten, werden zwei Möglichkeiten angeboten.

- Für LDAP-Server auf OpenLDAP-Basis besteht die Möglichkeit einer automatisierten Replizierung über das Syncrepl-Verfahren. **Dieses Verfahren wird favorisiert**, da es bereits eine automatische Prüfung des Datenbestands auf Vollständigkeit beinhaltet.
- Werden andere Systeme als OpenLDAP eingesetzt, so stehen LDIF-Dateien nach [RFC 2849] zum Download zur Verfügung. Dabei wird wöchentlich in einer LDIF-Datei der komplette Inhalt der LDAP-Datenbank zur Verfügung gestellt und über tägliche LDIF-Dateien die Änderungen inkrementell zum letzten Stand gepflegt. Die täglichen LDIF-Updates beinhalten Anweisungen zum Hinzufügen und Löschen von Zertifikaten. Die Aktualisierung eines eigenen LDAP-Servers ist als Holschuld festgelegt.

Die Annahmestellen haben darauf zu achten, dass vor dem Einlesen einer LDIF-Datei der strukturelle Aufbau ihres LDAP Servers mit dem des zentralen LDAP Servers übereinstimmt. Dies bedeutet, dass alle verwendeten Objekte und Attribute auf dem gespiegelten LDAP-Server mit denselben Datentypen vorhanden sein müssen.

Die Datenbank mit den Benutzern und Zugriffsrechten wird nicht als LDIF-Datei angeboten.

9.10 Betrieb des LDAP Servers

Jedes Trust Center, das am Datenaustausch-Verfahren teilnimmt, stellt einen eigenen LDAP Dienst zu Verfügung der von dem jeweiligen Trust Center eigenständig betrieben und gepflegt wird.

Die Trust Center vereinbaren untereinander eine automatisierte Replizierung.

9.11 Eindeutiger LDAP-Server zum Abruf aller Zertifikate

Der LDAP-Server des ITSG Trust Center stellt zentral alle Zertifikate der Datenaustausch Teilnehmer zum Abruf bereit. So ist gewährleistet, dass die Teilnehmer nur bei einer LDAP-Adresse anfragen müssen, bzw. die Replizierung einrichten müssen.

10 Anhang

10.1 Literaturverzeichnis

- [FIPS180-3] Federal Information Processing Standards (FIPS PUB) 180-3: Secure Hash Standard; October 2008
- [IMTTP0] Common ISIS-MailTrusT Specifications für Interoperable PKI Applications; ISIS-MTT Specification; Introduction; V1.0.2
- [IMTTP1] Common ISIS-MailTrusT Specifications für Interoperable PKI Applications; ISIS-MTT Specification; Part1: Certificate and CRL Profiles; V1.0.2
- [IMTTP2] Common ISIS-MailTrusT Specifications für Interoperable PKI Applications; ISIS-MTT Specification; Part2: PKI Management; V1.0.2
- [IMTTP3] Common ISIS-MailTrusT Specifications für Interoperable PKI Applications; ISIS-MTT Specification; Part3: Message Formats; V1.0.2
- [IMTTP4] Common ISIS-MailTrusT Specifications für Interoperable PKI Applications; ISIS-MTT Specification; Part4: Operational Protocols; V1.1
- [IMTTP6] Common ISIS-MailTrusT Specifications für Interoperable PKI Applications; ISIS-MTT Specification; Part6: Cryptographic Algorithms; V1.0.2
- [IMTTP6x] Common ISIS-MailTrusT Specifications für Interoperable PKI Applications; ISIS-MTT Specification; Part6: Cryptographic Algorithms; V1.1
- [PKCS#1] RSA Laboratories: PKCS #1 v.2.0: RSA Cryptography Standard, RSA Laboratories, Oktober 1998
- [PKCS#7] RSA Laboratories: PKCS #7: Cryptographic Message Syntax Standard; An RSA Laboratories Technical Note; Version 1.5; Revised November 1, 1993; <http://www.rsasecurity.com/rsalabs/pkcs/pkcs-7/index.html>; vgl. [RFC 2315].
- [PKCS#10] RSA Laboratories: PKCS #10 v1.7: Certification Request Syntax Standard; May 26, 2000; <http://www.rsasecurity.com/rsalabs/pkcs/pkcs-10/> vgl. [RFC 2314]
- [RFC 822] David H. Crocker: Standard for the Format of ARPA Internet Text Messages: Message Encryption and Authentication; August 1982
- [RFC 1422] S. Kent: Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management; IETF; RFC1422; Februar 1993.
- [RFC 1959 96] T. Howes, M. Smith: An LDAP URL Format; Juni 1996
- [RFC 2251] Lightweight Directory Access Protocol (v3). M. Wahl, T. Howes, S. Kille. December 1997
- [RFC 2279 98] UTF-8, a transformation format of ISO 10646; Januar 1998
- [RFC 2315] PKCS #7: Cryptographic Message Syntax Version 1.5. B. Kaliski. March 1998. (Format: TXT=69679 bytes) (Status: INFORMATIONAL)
- [RFC 2630] Cryptographic Message Syntax, R. Housley, June 1999
- [RFC 2830] Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security. J. Hodges, R. Morgan, M. Wahl. May 2000

- [RFC 2849] The LDAP Data Interchange Format (LDIF) - Technical Specification. G. Good. June 2000.
- [RFC 2986] PKCS #10: Certification Request Syntax Version 1.7. M. Nystrom, B. Kaliski. November 2000. (Format: TXT=27794 bytes) (Obsoletes RFC2314) (Status: INFORMATIONAL)
- [RFC 3565] Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS), J. Schaad, July 2003
- [SecSchnitt] „Security Schnittstelle für das Gesundheitswesen“ V2.1.0 – Stand: 10.10.2013

10.2 Abkürzungsverzeichnis

CA	Certification Authority
CMS	Cryptographic Message Syntax
LDAP	Lightweight Directory Access Protocol
OID	Object Identifier, Objektbezeichner
PCA	Policy CA
PKCS	Public Key Cryptography Standards
PKCS#1	RSA Cryptography Standard
PKCS#3	Diffie-Hellmann Key Agreement
PKCS#5	Password-based Encryption Standard
PKCS#6	Extended-Certificate Syntax Standard
PKCS#7	Cryptographic Message Syntax Standard
PKCS#8	Private-Key Information Syntax Standard
PKCS#9	Selected Attribute Types
PKCS#10	Certification Request Standard
PKCS#11	Cryptographic Token Interface (cryptoki)
PKCS#12	Personal Information Exchange Syntax Standard
*.p7b	Datei-Endung für PKCS#7-Zertifikat
*.p7c	Datei-Endung für PKCS#7-Zertifikat
*.p7m	Datei-Endung für PKCS#7 MIME-Nachricht (oft mit eingebettetem Original-Dokument)
*.p7s	Datei-Endung für PKCS#7-Signatur