Inhaltsverzeichnis

1. Grundlagen

Generelle Forderungen an die Sicherheitsdienste sind die Vertraulichkeit, die Integrität, die Authentifikation sowie die Verbindlichkeit der Kommunikation.

Als grundlegendes Verfahren der Sicherheitstechnologie nimmt die kryptographische Technik die zentrale Rolle auch in der angestrebten Public-Key-Infrastruktur (PKI) ein. Die Verfahren sollen folgenden Ansprüchen genügen, bzw. die ersten Voraussetzungen dafür schaffen:

- RSA-Algorithmus mit mindestens 4096 Bit Schlüssellänge; bis zur vollständigen Umstellung noch 2048 Bit Schlüssellänge
- AES – Advanced Encryption Standard [RFC 3565] (256 Bit, CBC-Modus)
- SHA-256 Algorithmus (256 Bit)
- X.509 v3-Zertifikate (Common PKI 2.02)
- lokale oder zentrale Generierung von Schlüsseln
- PKCS#7 – elektronische Signatur und Verschlüsselung
- Verzeichnisdienst
- Transportprotokoll (LDAP v3)

Die in diesem Dokument festgelegten Verschlüsselungs- und Hashfunktionen bieten nach heutigem Kenntnisstand eine ausreichende Sicherheit bis zum Jahr 2026. Die Veröffentlichungen des BSI (insbesondere [BSI-TR02102] fordern diese Algorithmen ab dem Jahr 2024 und empfehlen sie für 2023.

1.1 Einleitung

Die folgende Definition einer Security Schnittstelle ist als festgeschriebene, jedoch offengelegte Schnittstelle für das Gesundheits- und Sozialwesen gedacht.

Ziel der folgenden Definitionen ist es, im Gesundheits- und Sozialwesen eine gesicherte digitale Kommunikation unabhängig von der Art der jeweiligen Systeme zu gewährleisten.

Die Konzeption ist bei Bedarf als Migrationsstrategie für die bestehenden Sicherheitsverfahren ausgelegt. Die Migrationsansätze sehen vor, dass die vorhandenen Anwendungen für einen bestimmten Zeitrahmen weiter genutzt werden können. Die Beteiligten sollen den Einsatzzeitpunkt für die modifizierten Applikationen, soweit keine unabdingbare Notwendigkeit gegeben ist, selbst bestimmen können. Dementsprechend soll sowohl die heute bestehende, als auch die sich aus den nachfolgenden Definitionen ergebende Security-Technologie bis zum Zeitablauf der im Umlauf befindlichen Teilnehmer-Schlüssel parallel eingesetzt werden können (z.B. RSA-Schlüssellänge 2048 Bit neben RSA-Schlüssellänge 4096 Bit).

Aufgrund des bereits erwähnten hohen Bedürfnisses nach einer Sicherung der elektronischen Prozesse, verläuft die technologische Weiterentwickelung von PKI-Produkten rasant. Die in diesem Papier beschriebenen notwendigen Modifikationen sollen –wie in der Vergangenheit auch- im Sinne einer Migrations-Strategie in einem Phasenmodell erfolgen. Damit soll u. a. erreicht werden, dass die laufenden Prozesse nicht beeinträchtigt werden.

Die Sicherheit eines Kryptogramms –der verschlüsselten Inhalte- hängt wesentlich von der Länge der zur Erzeugung verwendeten Schlüssel ab. Dazu werden technische Richtlinien des BSI herausgegeben, die gemäß §95 SGB IV für diese Anlage verbindlich sind.

Um den Empfehlungen des BSI zu entsprechen, wurde in einer Migration auf eine Schlüssellänge von 4096 Bit und bei Signaturverfahren auf den Algorithmus RSASSA-PSS sowie bei Verschlüsselungsverfahren für den Nachrichtenschlüssel EME-OAEP (RSAES-OAEP) umgestellt.
Bis zum Ablauf der Gültigkeit der vorhandenen Zertifikate wird die Koexistenz und soweit erforderlich, die Interoperabilität der vorhandenen und der „neuen" Systeme vorausgesetzt.

Die Migrationsschritte für die Umstellung der Schlüssellänge und neuen Algorithmen stellen sich im Einzelnen wie folgt dar:

- Phase 1: Testphase für koordinierte Tests in 2018
  Ende 2017 wurden parallel zusätzlich eine PCA und CA's von den Trustcentern der ITSG und DKTIG für den neuen Signaturalgorithmus und größere Schlüssellänge erstellt. Die neuen CA's haben zunächst in der Testphase unveröffentlichte Testzertifikate für die Testteilnehmer bereitgestellt. An den koordinierten Interoperabilitätsstests haben Datenannahmestellen und registrierte Testteilnehmer teilgenommen.
  Alle weiteren Softwareersteller können in der Übergangsphase eigene, nicht koordinierte Tests mit Datenannahmestellen durchführen.

- Phase 2: Übergangsphase in 2019
  Ab Anfang 2019 konnten erste Teilnehmer produktive Zertifikate nach den neuen Verfahren über den offiziellen Antragsprozess erhalten. Dies setzte voraus, dass die Teilnehmer eine neue Softwareversion im Einsatz hatten, in der die neuen Verfahren bereits angewendet wurden. Seit der Ausgabe der ersten neuen Zertifikate mit 4096 Bit RSA-Schlüssel werden hierzu zusätzliche Schlüssellisten veröffentlicht (siehe 4.6.1.1). Gleichzeitig wurden die kryptografischen Prozesse bei der Nutzung neuer Zertifikate mit 4096 Bit RSA-Schlüssel auch an die Forderungen des BSI angepasst, so bei Signaturverfahren nur noch der Algorithmus RSASSA-PSS und bei Verschlüsselungsverfahren den Nachrichtenschlüssel EME-OAEP (RSAES-OAEP) verwendet werden. Die Datenannahmestellen unterstützen im gesamten Migrationszeitraum das alte und neue Verfahren unter Berücksichtigung der jeweils verwendeten Zertifikate der Teilnehmer mit 2048 oder 4096 Bit Schlüssellänge. Die Teilnehmer sollen die jeweilige öffentliche Schlüsselliste mit 2048 oder 4096 Bit Schlüssellänge

gemäß einem eingesetzten Zertifikat verwenden, um eine Vermischung der Algorithmen zwischen alten und neuen Verfahren in einer Datenlieferung zu vermeiden.

- Phase 3: Auslaufen des alten Verfahrens ab 2020
  Seit Anfang 2020 können Teilnehmer ausschließlich Zertifikate nach den neuen Verfahren erhalten und Zertifizierungsanträge nach den alten Verfahren werden abgewiesen. Ab diesem Termin sind über dem langen Migrationszeitraum die Teilnehmer betroffen, die mit Ablauf der Gültigkeit Ihrer Zertifikate ein neues Zertifikat beantragen wollen. Die bereits ausgestellten Zertifikate nach den alten Verfahren behalten jedoch ihre Gültigkeit und sind daher noch bis spätestens Ende 2022 gültig. Auch mit den bereits verwendeten Softwareprodukten können die Teilnehmer weiterhin signierte und verschlüsselte Daten nach den alten Verfahren bis Ende 2022 austauschen. Zum Migrationsende Anfang 2023 werden signierte und verschlüsselte Daten nach den alten Verfahren abgewiesen und somit müssen alle Teilnehmer spätestens bis dahin eine neue Softwareversion im Einsatz haben. Den Softwareersteller wurde ein Rollout einer neuen Software-Version in der Übergangsphase empfohlen, um eine Abweisung von Zertifizierungsanträgen ab Anfang 2020 zu vermeiden.

Der gesamte Umstellungsprozess ist somit spätestens am 31.12.2022 abgeschlossen. Es werden dann nur noch Zertifikate genutzt, denen ein RSA-Schlüssel mit einer Schlüssellänge von 4096 Bit zu Grunde liegt und gleichzeitig wird den Forderungen des BSI Rechnung getragen, dass bei Signaturverfahren nur noch den Algorithmus RSASSA-PSS und bei Verschlüsselungsverfahren für den Nachrichtenschlüssel EME-OAEP (RSAES-OAEP) verwendet werden.

Weiterhin wurden die Registrierungsprozesse in den beteiligten Trust Centern an die technischen Richtlinien des BSI angepasst, damit die Vorgaben eines substanziellen Schutzbedarfs erfüllt werden.

1.2 Grundlagen

1.2.1 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 absolut gültige 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.

1.2.2 Definitionen

Anhand des Standards PKSC#10 wird ein Profil erarbeitet. 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 5.8 beschrieben.

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 technische Profil für X.509v3-Zertifikate ist im Kapitel 4.4 beschrieben.

Dieses gilt insbesondere für die Vorgaben hinsichtlich des Signaturgesetzes im Abschnitt 1.1 „Einleitung“, die Forderungen nach qualifizierten Zertifikaten mit Anbieterakkreditierung werden derzeit nicht gestellt. Hinsichtlich der digitalen Umschläge für PKCS#10-Zertifizierungsanfragen und für PKCS#10-Zertifizierunganfragen 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.

1.3 Vorbemerkungen

Es werden derzeit auf dem Markt verschiedene Verfahren für die Generierung von Sicherheitsfunktionen angeboten. Die bedeutenden Verfahren haben im Allgemeinen gleiche Konstruktionsmerkmale, unterscheiden sich aber in einigen Details.

Grundlage sind die Common PKI 2.0 Spezifikationen für das PKI-Management, welche sich im Wesentlichen an den Spezifikationen des PKIX-Dokumentes „Certificate Management Protokolls [PKIX-

CMP 98] orientieren. Da kein „neues“ Verfahren dem Sinne nach konzipiert wird, wurde auf vorhandene Funktionalitäten aufgebaut und das Fachwissen am Markt etablierter Hersteller/Anbieter von Security-Software und Tele Trust Deutschland e. V. integriert.

Grundlage des Schlüsselmanagements ist die Verwendung von Zertifikaten, um öffentliche Schlüsselinformationen authentisch dem Sender und dem Empfänger einer Nachricht zur Verfügung zu stellen.

Derzeitige Basis des Ansatzes bilden dabei die in den RFC1421 - RFC1424 festgelegten Grundlagen zu Electronic Mail.

Die aufgeführten Spezifikationen orientieren sich an etablierten Standards (z. B. X.509 und PKCS#7). Die im Einsatz befindlichen Verfahren sind im Rahmen der Migration entsprechend zu berücksichtigen. Die Spezifikationen beschreiben die Minimalanforderungen, die zur Teilnahme am Verfahren zu erfüllen sind.

Das Management von Zertifikaten und Schlüsseln orientiert sich an Standards (Basis des Ansatzes bildet z. B. CMC – Certificate Management Messages over CMS) – siehe auch RFC 6402 bzw. RFC 5272. Diese Festlegung ist mit Hinblick auf die Trust Center-Betreiber sinnvoll. Damit ist die Beantragung von Zertifikaten über PKCS#10 und PKCS#7 neben CMS vorgesehen. Der ausschließliche Verweis auf CMS unterstellt, dass eine Vielzahl an Nachrichtentypen zu unterstützen wäre, die aber zum Teil für das Verfahren nicht bedeutend sind.

Die Zertifikatsformate X.509v3 werden entsprechend in der ASN.1-Syntax beschrieben. Erklärtes Ziel ist es, eine Absprache bezüglich einiger Parameter zu treffen, um für das Sicherheitsverfahren im Gesundheitswesen eine, auch im Hinblick auf die Zukunft, sichere Kommunikation zu gewährleisten. Dieses Ziel kann nur durch die Beteiligung der am Markt etablierten Anbieter und Hersteller von Sicherheitssoftware, die i. d. R. dem TeleTrust Deutschland e. V. angehören, erreicht werden. Die vorliegende Dokumentation folgt einem international konsolidierten Konzept für Informations- und Kommunikationssicherheit in offenen IT-Systemen. Die Auswahl der Algorithmen und Schlüsseldefinitionen, der Datenformate, der Zertifizierungsstruktur und der Struktur der Adressen für die Schnittstelle sind Grundlage dafür, dass

- die Interoperabilität von Anwendungen,
- die Interoperabilität zwischen Zertifizierungsinstanzen beim
- Schlüsselmanagement und anderen Trusted Third Party - Diensten,
- die Bereitstellung von miteinander kompatiblen Hard- und
- Softwarekomponenten und Diensten durch Technologie- und Produktentwickler

gewährleistet werden können.

1.4 Kommunikation zum Datenaustausch im Gesundheits- und Sozialwesen

Sowohl die Trust Center als auch die zur Verfügung stehenden Sicherheitssysteme decken derzeit die Anforderungen einer X.500 Implementierung nicht ab. Die Anzahl an Zertifikaten und auch die mittelfristig erwarteten Teilnehmerzahlen ermöglichen den kompletten Austausch der Verzeichnisse auf der Grundlage der heute verwendeten TCP/IP-Protokollfamilie mittels E-Mail, ftp oder http. Die Basis hierfür bildet die heutige Ausrichtung des Datenaustauschverfahrens innerhalb der Sozialversicherung.

Für den Aufruf von Zertifikaten und Sperrlisten wird ein Protokoll benötigt, welches Funktionalitäten bietet, wie z. B. das selektive Suchen in Verzeichnissen usw. Hier bietet LDAP (Ligthweight Directory Access Protocol) v3 [RFC 4510 - 4519] die geeigneten Funktionalitäten für den Verzeichnisabruf. Das Verzeichnis entspricht dem X.500 Modell. Zertifikate sind als Werte vom Typ „userCertificate“ codiert, die Sperrlisten sind vom Typ „certificateRevocationlist“ codiert. Die Zertifikate und Sperrlisten sind im Verzeichnis in ihrer BER-codierten Form als Binärdateien gespeichert. Bei der Nutzung von LDAP sind in Anlehnung an die MailTrust (Common-PKI 2.0) Spezifikationen mindestens die Operationen: „bind“, „search“ und „unbind“ vorgesehen. Dabei sind in Anlehnung an die aktuell gültigen MailTrust Spezifikationen die diesbezüglichen Vorgaben zu den Austauschformaten zu berücksichtigen.

Als Datei-Extensions sind anzugeben:

Revocation List	.crl
Zertifizierungsantwort	.p7c
Zertifizierungsanfrage	.p10

1.5 Verbindlichkeit

Die in diesem Dokument getroffenen Spezifikationen haben einen verbindlichen Charakter. Wird bei der Datenannahme ein Abweichen von diesen Spezifikationen festgestellt, so führt dies zur Abweisung der Datenlieferung.

2. Kryptografische Algorithmen und Datenformate

In diesem Kapitel werden die zulässigen kryptografischen Algorithmen, die einzuhaltenden Schlüssellängen sowie die grundlegenden Datenformate für Zertifikate beschrieben. Auf Basis der damit gebildeten Sicherheitsinfrastruktur werden in den folgenden Kapiteln der Nachrichtenaustausch und das Schlüsselmanagement aufgesetzt.

2.1 Krypto-Algorithmen

Die Auswahl der kryptografischen Algorithmen ist an [CommonPKI-6] angelehnt und folgt daher auch der dort vorgenommenen Einteilung in verschiedene Gruppen je nach Verwendungszweck, wobei die einzelnen Algorithmen dann in den Unterabschnitten ggf. auch mehrfach genannt werden. Die Algorithmen dienen als Grundbausteine für eine technische Sicherheitsinfrastruktur, in

welcher die eingangs genannten Schutzziele, d.h. die Gewährleistung von Vertraulichkeit, Integrität und Authentizität ausgetauschter Informationen im Gesundheits- und Sozialwesen, schließlich erreicht werden sollen.

Neben der dominierenden Frage hinsichtlich der sicherheitstechnischen Eignung bestimmter Verfahren (vgl. z.B. Ausführungen in [BNA-AlgKat] oder [BSI-TR02102]) muss auch die Effizienz des Datenaustausches in der Risikoabwägung beachtet werden. Dies betrifft sowohl die Algorithmen als auch deren spezifische Parameter (Schlüssellängen, Betriebsmodi etc.) bzw. ihre Einbettung im Gesamtsystem. Beispielsweise wird durch die Kombination von Secret-Key-Verfahren und Public-Key-Verfahren in Form eines hybriden Verschlüsselungssystems einerseits die aufgewandte Rechenzeit verkürzt und andererseits stehen auch die Vorteile einer effizienten Schlüsselverteilung zur Verfügung.

2.1.1 Einweg-Hashfunktionen (One-Way Hash Functions)

Als Hashfunktionen ist ausschließlich der vom NIST in [FIPS180-3] definierte Algorithmus SHA-256 zu verwenden.

Name	OID
SHA-256	2.16.840.1.101.3.4.2.1

Die Einweg-Hashfunktionen werden dazu verwendet, aus einer gegebenen Nachricht einen so genannten „Message Digest“ zu errechnen. Ein „Message Digest“ ist dabei ein Wert, der weitgehend eindeutig einer Nachricht zugeordnet werden kann (Kollisionsresistenz der Hashfunktion). Aus dem „Message Digest“ lässt sich jedoch die zugrundeliegende Nachricht nicht wieder zurückberechnen (Einweg-Eigenschaft der Hashfunktion).

- Eingabe: die zu verarbeitende Nachricht
- Ausgabe: eine Zeichenfolge (Hashwert) fester Bit Länge

Die Ausgabe der jeweiligen Hashfunktion wird nicht explizit in Zertifikaten oder ausgetauschten Nachrichten angegeben. Sie wird jedoch u.a. bei der Signaturerstellung verwendet und hat damit Auswirkungen auf die Schutzziele Integrität und Authentizität. Der erzeugte Hashwert hat bei SHA-256 eine Länge von 256 Bit. Bei der Implementierung ist darauf zu achten, dass jeweils der durch den oben angegebenen Objektbezeichner (OID) referenzierte Algorithmus gemäß den Vorgaben in [FIPS180-3] verwendet wird.

Profilierung:
In der vorliegenden Spezifikation werden Werte von Hashfunktionen sowohl bei der Signaturerstellung als auch für 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 beigefügt.

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.

2.1.2 Signaturalgorithmen (Signature Algorithms)

Bis zum Migrationsende Ende 2022 ist alternativ als Signaturalgorithmus Sha256withRSAEncryption gemäß PKCS#1 bzw. RFC 8017 mit SHA-256 als Hashfunktion (vgl. Abschnitt 2.1.1) ohne Parameter mit einer RSA-Schlüssellänge von 2048 Bit zulässig.
Mit der Ausgabe von Zertifikaten mit einer RSA-Schlüssellänge von 4096 Bit änderte sich der Signaturalgorithmus. Statt des bisherigen RSA-Verfahrens nach PKCS#1 v1.5 findet das Verfahren Signature Scheme with Appendix PSS" (üblicherweise mit „RSASSA-PSS" abgekürzt) gemäß RFC 8017 mit nachfolgenden Parameter Verwendung. Im Verfahren „RSASSA-PSS" ist eine RSA-Schlüssellänge von 2048 Bit nicht mehr zulässig und die Migrationsstrategie gemäß Abschnitt 1.1 zu beachten.

Name	OID
id-RSAES-PSS	1.2.840.113549.1.1.10

Es wird als öffentlicher Exponent „e" die 4. Fermatsche Zahl (0x10001 bzw. 2^16 + 1 = 65537 dezimal) verwendet.

Dies betrifft die Erstellung und Validierung von Signaturen

- von Zertifikaten,
- der PKCS#10-Zertifizierungsanfragen,
- der PKCS#7-Zertififizierungsantworten,
- der eigentlichen Nachrichten im Nachrichtenaustausch

Der Signaturalgorithmus wird zum elektronischen Signieren von Daten verwendet, um sie einerseits gegen unbefugte Veränderung zu schützen (Integrität) sowie andererseits ihre Herkunft nachzuweisen (Authentizität). Er ist eine Kombination aus einer Hashfunktion und einem Public-Key-Verfahren (öffentlicher und privater Schlüssel).

- Eingabe: die zu signierende Nachricht, der private Schlüssel des Signierenden
- Ausgabe:die digitale Signatur

Der zugehörige Objektbezeichner (OID) aus der obigen Tabelle wird im Feld Algorithm der Struktur Signature eines Zertifikates oder eines PKCS#10-Requests eingetragen.

2.1.2.1 Datenfelder zur Verwendung des Signaturalgorithmus RSAES-PSS

Anders als bei der bisher verwendeten Angabe von sha256WithRSAEncryption, wird der im RSASSA-PSS-Verfahren verwendete Hashalgorithmus und weitere Werte über entsprechende Parameter mittels RSASSA-PSS-params definiert. Deren ASN1-Struktur mit den Default-Werten ist wie folgt definiert:

```
RSASSA-PSS-params ::= SEQUENCE {
   hashAlgorithm      [0] HashAlgorithm      DEFAULT sha1,
   maskGenAlgorithm   [1] MaskGenAlgorithm   DEFAULT mgf1SHA1,
   saltLength         [2] INTEGER            DEFAULT 20,
   trailerField       [3] TrailerField       DEFAULT trailerFieldBC
}
```

Im Folgenden werden die Parameter-Werte für die Verwendung von SHA256 als Signaturverfahren erläutert.

2.1.2.1.1 Datenfeld HashAlgorithm

Das Feld HashAlgorithm gibt den zu verwendenden Hash-Algorithmus an. Da SHA1 als Default-Wert nicht unterstützt wird, ist die explizite Angabe des Parameter-Wertes für die Verwendung von SHA256 als Signaturverfahren erforderlich. Nach [RFC-8017] ist die ASN1-Struktur für das Feld, das den Signaturalgorithmus angibt, wie folgt definiert:

```
HashAlgorithm ::= AlgorithmIdentifier {
    {OAEP-PSSDigestAlgorithms}
}

OAEP-PSSDigestAlgorithms    ALGORITHM-IDENTIFIER ::= {
    { OID id-sha1       PARAMETERS NULL }|
    { OID id-sha224     PARAMETERS NULL }|
    { OID id-sha256     PARAMETERS NULL }|
    { OID id-sha384     PARAMETERS NULL }|
    { OID id-sha512     PARAMETERS NULL }|
    { OID id-sha512-224 PARAMETERS NULL }|
    { OID id-sha512-256 PARAMETERS NULL },
    ...  -- Allows for future expansion --
}
```

Für die Angabe der Hashfunktion SHA256 ist die OID gemäß Kapitel 2.1.1 Einweg-Hashfunktionen zu verwenden.

```
sha256    HashAlgorithm ::= {
    algorithm   id-sha256,
    parameters  SHA256Parameters : NULL
```

```
}
```

2.1.2.1.2 Datenfeld MaskGenAlgorithm

Das Feld MaskGenAlgorithm definiert das Formatierungsverfahren. RFC-8017 sieht aktuell nur das Verfahren MGF1 vor.

```
MaskGenAlgorithm ::= AlgorithmIdentifier { {PKCS1MGFAlgorithms} }

PKCS1MGFAlgorithms    ALGORITHM-IDENTIFIER ::= {
    { OID id-mgf1 PARAMETERS HashAlgorithm },
    ...  -- Allows for future expansion --
}

id-mgf1    OBJECT IDENTIFIER ::= { pkcs-1 8 }
```

Name	OID
id-mgf1	1.2.840.113549.1.1.8

Das Formatierungsverfahren ist generisch und basiert wiederrum auf einem Hash-Verfahren, dass entsprechend der Hash-Funktion im Feld HashAlgorithm zu wählen ist. Daraus ergibt sich für die Verwendung von SHA256 folgende Parameter-Definition:

```
mgf1SHA256    MaskGenAlgorithm ::= {
    algorithm   id-mgf1,
    parameters  HashAlgorithm : sha256
}
```

Für die Angabe der Hashfunktion SHA256 ist die OID gemäß Kapitel 2.1.1 Einweg-Hashfunktionen zu verwenden.

2.1.2.1.3 Datenfeld saltLength

Das Feld saltLength gibt die Länge des zu verwendenden Salt-Wertes an und ist in RFC-8017 definiert. Gemäß der Empfehlung des BSI wird bei SHA256 eine Salt-Länge 32 verwendet.

2.1.2.1.4 Datenfeld TrailerField

Für das Feld TrailerField ist gemäß RFC-8017 ein fester Wert vorgesehen:

```
TrailerField ::= INTEGER { trailerFieldBC(1) }
```

2.1.3 Verschlüsselungsalgorithmen für Daten (Content Encryption Algorithmus)

Als Verschlüsselungsalgorithmus für Daten ist standardmäßig der AES-Algorithmus [RFC 3565] mit 256 Bit Schlüssellänge und CBC-Betriebsmodus (id-aes256-CBC) zu verwenden.

Name	OID
id-aes256-CBC	2.16.840.1.101.3.4.1.42

Der symmetrische Verschlüsselungsalgorithmus (Secret-Key-Verfahren) ist der eigentliche Algorithmus mit dem die Daten ver- und entschlüsselt werden, um die Vertraulichkeit der ausgetauschten Informationen zu schützen. Hierzu müssen Verschlüsseler und Entschlüsseler über den gleichen Nachrichtenschlüssel (auch Session-Key genannt) verfügen, der wiederum zufällig und individuell für jede Nachricht gewählt werden sollte.

	Verschlüsselung	Entschlüsselung
• Eingabe:	die zu verschlüsselnden Daten der Nachrichtenschlüssel	die verschlüsselten Daten der Nachrichtenschlüssel
• Ausgabe:	die verschlüsselten Daten	die entschlüsselten Daten

Der Nachrichtenschlüssel ist der Schlüssel, mit dem die Daten verschlüsselt werden. Er wird für jede Nachricht jeweils zufällig bestimmt.

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

2.1.4 Verschlüsselungsalgorithmen für den Nachrichtenschlüssel (Key Encryption Algorithmus)

Zur Verschlüsselung des Nachrichtenschlüssels ist bis zum Migrationsende Ende 2022 alternativ der rsaEncryption-Algorithmus gemäß PKCS#1 bzw. RFC 8017 ohne Parameter mit einer RSA-Schlüssellänge von 2048 Bit zulässig.
Mit der Ausgabe von Zertifikaten mit einer RSA-Schlüssellänge von 4096 Bit änderte sich der Verschlüsselungsalgorithmus für den Nachrichtenschlüssel. Statt des bisherigen RSA-Verfahren nach PKCS#1 v1.5 findet das Verfahren „EME-OAEP“ (bei Verwendung mit RSA auch als RSAES-OAEP bezeichnet) gemäß RFC 8017 mit nachfolgenden Parameter Verwendung. Im Verfahren „EME-OAEP“ ist eine RSA-Schlüssellänge von 2048 Bit nicht mehr zulässig und die Migrationsstrategie gemäß Abschnitt 1.1 zu beachten.

Name	OID
id-RSAES-OAEP	1.2.840.113549.1.1.7

Es wird als öffentlicher Exponent „e“ die 4. Fermatsche Zahl (0x10001 bzw. 2^16 + 1 = 65537 dezimal) festgelegt.

Die Verschlüsselung des Nachrichtenschlüssels erfolgt mit Hilfe eines asymmetrischen Verfahrens, damit die Schlüsselverteilung über eine Public-Key-Infrastruktur (PKI) möglich ist. Hierfür gibt es ein Schlüsselpaar (auch Interchange-Key genannt) bestehend aus öffentlichem und privaten Schlüssel für jeden Teilnehmer.

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

Für den oben angegebenen RSA-Algorithmus ist hier eine Schlüssellänge von 4096 Bit zu verwenden. Für einen Übergangszeitraum ist eine Schlüssellänge von 2048 Bit zulässig. Dabei ist die Migrationsstrategie gemäß Abschnitt 1.1 zu beachten. Der zugehörige Objektbezeichner wird im Feld Algorithm der Struktur subjectPublicKeyInfo eines Zertifikates oder PKCS#10-Requests eingetragen.

2.1.4.1 Datenfelder zur Verwendung des Verschlüsselungsalgorithmus RSAES-OAEP

Anders als bei der bisher verwendeten Angabe von rsaEncryption, werden im OAEP-Verfahren einige verfahrensspezifische Werte über entsprechende Parameter mittels RSAES-OAEP-params definiert. Dabei handelt es sich um die Angabe eines Hash-Verfahrens, eines Formatierungsverfahrens sowie eines Labels. Deren ASN1-Struktur mit den Default-Werten ist wie folgt definiert:

```
RSAES-OAEP-params ::= SEQUENCE {
   hashAlgorithm      [0] HashAlgorithm     DEFAULT sha1,
   maskGenAlgorithm   [1] MaskGenAlgorithm  DEFAULT mgf1SHA1,
   pSourceAlgorithm   [2] PSourceAlgorithm  DEFAULT pSpecifiedEmpty
```

Im Folgenden werden die Parameter-Werte für die Verwendung von SHA256 als Hashverfahren erläutert.

2.1.4.1.1 Datenfeld HashAlgorithm

Das Feld HashAlgorithm gibt den zu verwendenden Hash-Algorithmus an. Da SHA1 als Default-Wert nicht unterstützt wird, ist die explizite Angabe des Parameter-Wertes für die Verwendung von SHA256 als Hashverfahren erforderlich.

```
HashAlgorithm ::= AlgorithmIdentifier {
   {OAEP-PSSDigestAlgorithms}
}

OAEP-PSSDigestAlgorithms    ALGORITHM-IDENTIFIER ::= {
   { OID id-sha1       PARAMETERS NULL }|
   { OID id-sha224     PARAMETERS NULL }|
   { OID id-sha256     PARAMETERS NULL }|
   { OID id-sha384     PARAMETERS NULL }|
   { OID id-sha512     PARAMETERS NULL }|
```

```
   { OID id-sha512-224 PARAMETERS NULL }|
   { OID id-sha512-256 PARAMETERS NULL },
   ...  -- Allows for future expansion --
}
```

Für die Angabe der Hashfunktion SHA256 ist die OID gemäß Kapitel 2.1.1zu verwenden.

```
sha256    HashAlgorithm ::= {
   algorithm   id-sha256,
   parameters  SHA256Parameters : NULL
}
```

2.1.4.1.2 Datenfeld MaskGenAlgorithm

Das Feld MaskGenAlgorithm definiert das Formatierungsverfahren. [RFC-8017] sieht aktuell nur das Verfahren MGF1 vor.

```
MaskGenAlgorithm ::= AlgorithmIdentifier { {PKCS1MGFAlgorithms} }

PKCS1MGFAlgorithms    ALGORITHM-IDENTIFIER ::= {
   { OID id-mgf1 PARAMETERS HashAlgorithm },
   ...  -- Allows for future expansion --
}

id-mgf1    OBJECT IDENTIFIER ::= { pkcs-1 8 }
```

Name	OID
id-mgf1	1.2.840.113549.1.1.8

Das Formatierungsverfahren ist generisch und basiert wiederrum auf einem Hash-Verfahren, dass entsprechend der Hash-Funktion im Feld HashAlgorithm zu wählen ist. Daraus ergibt sich für die Verwendung von SHA256 folgende Parameter-Definition:

```
mgf1SHA256    MaskGenAlgorithm ::= {
   algorithm   id-mgf1,
   parameters  HashAlgorithm : sha256
}
```

Für die Angabe der Hashfunktion SHA256 ist die OID gemäß Kapitel 2.1.1 Einweg-Hashfunktionen zu verwenden.

2.1.4.1.3 Datenfeld PSourceAlgorithm

Das Feld PSourceAlgorithm definiert die zu verwendenden Verschlüsselungsparameter. [RFC-8017] sieht aktuell nur die Angabe eines „Labels“ einen „Octet String“ mit festem Wert vor.

```
PSourceAlgorithm ::= AlgorithmIdentifier {
   {PKCS1PSourceAlgorithms}
}

PKCS1PSourceAlgorithms    ALGORITHM-IDENTIFIER ::= {
   { OID id-pSpecified PARAMETERS EncodingParameters },
   ...  -- Allows for future expansion --
}

id-pSpecified    OBJECT IDENTIFIER ::= { pkcs-1 9 }

EncodingParameters ::= OCTET STRING(SIZE(0..MAX))
```

Als Default-Wert für das Label ist in [RFC-8017] die Angabe eines leeren Labels vorgesehen:

```
pSpecifiedEmpty    PSourceAlgorithm ::= {
        algorithm   id-pSpecified,
        parameters  EncodingParameters : emptyString
     }

emptyString    EncodingParameters ::= "H
```

2.1.5 Algorithmen zur Nutzung des öffentlichen Schlüssels (Subject Public Key Algorithmus)

Auch zur weiteren Nutzung des öffentlichen Schlüssels (z.B. zur Signaturverifikation) wird RSA gemäß RFC 8017 eingesetzt. Mit folgender OID wird die zulässige Verwendung der Schlüssel gemäß RFC 4055 definiert:

Name	OID
rsaEncryption	1.2.840.113549.1.1.1

Als Schlüssellänge wird sowohl für Zertifizierungsstellen (PCA und CA) als auch für einfache Teilnehmer eine Länge von 4096 Bit vorgeschrieben. Für einen Übergangszeitraum ist eine Schlüssellänge von 2048 Bit zulässig. Dabei ist die Migrationsstrategie gemäß Kapitel 1.1 zu beachten. Für Zertifizierungsstellen kann die Schlüssellänge nach gesonderter Festlegung auch größer sein.

Je nach Anwendungsumfeld (Signatur oder Ver-/Entschlüsselung des Nachrichtenschlüssels) werden die Schlüssel im entsprechenden Algorithmus wie unter 2.1.2 oder 2.1.4 beschrieben verwendet.

Der zugehörige Objektbezeichner (OID) wird im Feld Algorithm der Struktur subjectPublicKeyInfo eines Zertifikates oder PKCS#10-Requests eingetragen (vgl. Abschnitt 4.4.8).

2.2 Datenformate

Die folgenden Spezifikationen und Datenformate sind vorgesehen.

2.2.1 Spezifikation: Session-Key

Als Session-Key ist standardmäßig der AES-Algorithmus mit 256 Bit Schüssellänge und CBC-Betriebsmodus (id-aes256-CBC) vorzusehen.

2.2.2 Spezifikation: Interchange Key

Als Interchange Key ist RSA mit den unten beschriebenen Parametern einzusetzen.

2.2.3 Spezifikation: Hashfunktion/Signaturalgorithmus

Als Hash-Funktion ist standardmäßig SHA-256 vorzusehen.
Die Hash-Funktion wird zum Erzeugen eines so genannten Message Digest verwendet, aus dem die elektronische Unterschrift gebildet wird.

2.2.4 Spezifikation: RSA Schlüssellänge

Die RSA Schlüssellänge beträgt:

- PCA-Schlüssel 4096 Bit (Standard); nach gesonderter Festlegung auch größer
- CA-Schlüssel 4096 Bit (Standard); nach gesonderter Festlegung auch größer
- Teilnehmer 4096 Bit (Standard)

Für einen Übergangszeitraum ist eine Schlüssellänge von 2048 Bit zulässig. Dabei ist die Migrationsstrategie gemäß Kapitel 1.1zu beachten.

2.2.5 Spezifikation: Öffentlicher Exponent des RSA Algorithmus

Als RSA Exponent soll die 4. Fermat Zahl ($2^{16}+1$) gewählt werden (siehe X.509).

2.2.6 Spezifikation: Schlüsselformat für den Public-Key

Hier ist die „Abstract Syntax Notation“ (ASN.1) sowie der Standard X.509 maßgeblich.

2.2.7 Spezifikation: Zertifikate

Es sind V3-Zertifikate gemäß X.509 (inklusive Extensions, soweit deren Unterstützung im Standard gefordert ist; siehe unten) zu verwenden. Die Komponenten/ Implementierungen sind so einzustellen/ anzulegen, dass Objektbezeichner (OIDs) für unbekannte Extensions in Zertifikaten ignoriert werden. Damit bleiben eine spätere Verfahrenserweiterung und die Nutzung spezifischer Extensions möglich.

Zertifikate sind in ASN.1 Syntax Notation sowie entsprechend der Norm X.509 zu implementieren. Bei der Codierung der Zertifikate sind die „Distinguished Encoding Rules“ (DER) entsprechend X.509 einzuhalten.

Die Extensions “SubjectKey-Identifier”, “AuthorityKey-Identifier”, “KeyUsage”, “CertificatesPolicies”, “SubjectAlternative-Name”, “BasicConstraints” und “CRLDistributions_Points“ müssen unterstützt werden.

Die Erweiterung „BasicConstraints“ dient der Erkennung von CA-Zertifikaten und umfasst die Felder „CA“ und „PathLenConstraint“: Das CA-Feld von „BasicContraints“ muss den Wert „TRUE“ (Teilnehmer darf Zertifikate signieren) enthalten. Das Feld „PathLenConstraint“ kann gesetzt sein. Wenn es angegeben ist, dann muss die Pfadlänge auf „0“ gesetzt werden, soweit die Zertifizierungsstelle ausschließlich Teilnehmerzertifikate (d.h. keine Zertifikate für andere CAs) ausstellen darf.

2.2.8 Spezifikation: PKCS#7 (Public Key Cryptography Standards)

Die allgemeine Syntax für Datenströme oder Dateien, die mit den kryptografischen Algorithmen aus Abschnitt 2.1 bearbeitet werden, ist gemäß PKCS#7 vorzunehmen.

2.3 Zufallszahlen

Modernen kryptographischen Verfahren liegt die Erzeugung von ausreichend sicheren Zufallszahlen zu Grunde. Ausreichend sicher bedeutet in diesem Zusammenhang insbesondere, dass diese ausreichend groß und ausreichend zufällig erzeugt und verwendet werden. Nach den Empfehlungen des BSI sollen Zufallszahlen zukünftig in der Regel mit einer Entropie von mindestens 120 Bit – statt bislang 100 Bit – erzeugt werden. Da das Generieren von ausreichend sicheren Zufallszahlen insbesondere bei der Erzeugung der (privaten und öffentlichen sowie symmetrischen) Schlüssel erforderlich ist, ist diese Anforderung insbesondere auf Seiten der einzelnen Teilnehmer des Verfahrens relevant und entsprechend zu berücksichtigen.

3. Nachrichtenaustausch

Für den Datenaustausch im Sinne dieser Richtlinie sind insgesamt drei verschiedene Nachrichtentypen nach PKI-Nachrichten und Nachrichten für den Datenaustausch zu unterscheiden.

Nachrichten werden im Datenaustausch des Gesundheits- und Sozialwesen 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 Sozialversicherungsträgern durchgeführt werden kann. Der Nachrichtenaustausch findet zwischen den folgenden Beteiligten statt:

- Leistungserbringer und Sozialversicherungsträger zur Übermittlung der Abrechnungsdaten

oder

- Arbeitgebern zur Übermittlung von Meldungen zum Sozialversicherungsträger

oder

- Sozialversicherungsträger untereinander.

3.1 Signierte und verschlüsselte Nachrichten

Für den Datenaustausch zwischen Sozialversicherungsträgern und Leistungserbringern / Arbeitgebern / untereinander (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 [CommonPKI-3] 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.

3.2 Aufbau der PKCS#7-signierten und verschlüsselten Nachricht

Referenz: [CommonPKI-3] Tabelle 3.2
[CommonPKI-3] Tabelle 3.3,
[RFC 5652] Abschnitt 5
[RFC 5652] Abschnitt 6

Eine verschlüsselte Nachricht ist gemäß [CommonPKI-3] Abschnitt 3.3 vom Typ EnvelopedData, eine signierte gemäß [CommonPKI-3] Abschnitt 3.2 vom Typ SignedData. Die zu sichernden Daten werden wie in [CommonPKI-3] 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 Kapitel 3.2.1 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 3.2.2, wird das SignedData-Objekt als Input verwendet. Der Typ EnvelopedData besteht allgemein aus zu verschlüsselnden Daten und einem für einen oder mehrere Empfänger separat 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 empfängerspezifische Informationen werden in einem RecipientInfo-Wert zusammengefasst.
- Abschließend werden alle RecipientInfo-Werte mit den verschlüsselten Daten zum CMS-Objekt EnvelopedData zusammengesetzt.

3.2.1 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 }
```

3.2.1.1 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.

3.2.1.2 Datenfeld „digestAlgorithms“

Dieses Datenfeld enthält genau einen Objektbezeichner[1] für den Hash-Algorithmus, der zur Signatur der Daten eingesetzt wird.

[1] 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.

Als Hash-Algorithmen ist nur SHA-256 vorgesehen. Der zu verwendende Objektbezeichner ist von [CommonPKI-2] Tabelle 2 auf OID 2.16.840.1.101.3.4.2.1 (SHA-256) festgelegt; vgl. Abschnitt 2.1.1.

3.2.1.3 Datenfeld „encapContentInfo“

```
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.

3.2.1.4 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
```

Profilierung:
Verwendet werden ausschließlich X509-Zertifikate. Die Reihenfolge ist in [CommonPKI-2] nicht festgelegt und ist deshalb als Festlegungsvorschlag anzusehen.

Hier findet die Spezifikation der X509v3-Zertifikate Anwendung; vgl. Kapitel 4.4. Jedes der im PKCS#7-Datenobjekt unter der Datenstruktur SignedData im Feld Certificates anzugebende Zertifikat hat den im Kapitel 4.4 formulierten Anforderungen einschließlich der Profilierung an X509v3-Zertifikate zu genügen.

3.2.1.5 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.

3.2.1.6 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 }
```

3.2.1.6.1 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.

3.2.1.6.2 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.

3.2.1.6.3 Datenfeld „digestAlgorithm“

Das Feld digestAlgorithm enthält den Indentifikator des Hash-Algorithmus des Absenders.

Profilierung:
Siehe 2.1.1 Einweg-Hash-Funktionen (One-Way Hash Functions).

3.2.1.6.4 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 5652] beschrieben.

3.2.1.6.5 Datenfeld „signatureAlgorithm“

Das Feld signatureAlgorithm enthält den Indentifikator des Signatur Algorithmus des Absenders.

Profilierung:
Siehe 2.1.2 Signaturalgorithmen (Signature Algorithms).

3.2.1.6.6 Datenfeld „signature“

Das Feld signature enthält die digitale Signatur des Absenders.

3.2.1.6.7 Datenfeld „unsignedAttrs“

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

Profilierung:
Dieses Feld entfällt.

3.2.2 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
```

3.2.2.1 Datenfeld „version“

Referenz: [CommonPKI-3] Tabelle 6#1,
[RFC 5652] 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 [CommonPKI-3] 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.

3.2.2.2 Datenfeld „originatorInfo“

Referenz: [CommonPKI-3] 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 3.2.2

3.2.2.3 Datenfeld „RecipientInfos“

Referenz: [CommonPKI-3] Tabellen 6 und 7
[RFC 5652], Abschnitt 6.2

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

Vgl. [CommonPKI-3] Tabelle 7 und [RFC 5652], Abschnitt 6.2:

```
RecipientInfos ::= SET OF RecipientInfo
RecipientInfo ::= CHOICE {
ktri KeyTransRecipientInfo,
kari [1] KeyAgreeRecipientInfo,
```

kekri [2] KEKRecipientInfo }

Wie in [CommonPKI-3] wird ausschließlich die Datenstruktur „KeyTransRecipientInfo“ verwendet. Vgl. [CommonPKI-3] 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 [CommonPKI-3] 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.

3.2.2.3.1 Teildatenfeld „version“

Referenz: [CommonPKI-3] Tabelle 8.#1
[RFC 5652], 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. [CommonPKI-3] Tabelle 8, Notes [1]. Entsprechend Kapitel 6.2.1 [RFC 5652] ist dies der Fall, wenn die Variante issuerAndSerialNumber gewählt wird. (s. u.)

3.2.2.3.2 Datenfeld „rid“

Referenz: [CommonPKI-3] Tabelle 8.#2
[RFC 5652], 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. [CommonPKI-3] 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.

3.2.2.3.3 Teildatenfeld „keyEncryptionAlgorithm“

Referenz: [CommonPKI-3] Tabelle 8.#3
[RFC 5652], 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 RSA-OAEP vorgesehen; vgl. Abschnitt 2.1.4.

3.2.2.3.4 Teildatenfeld „encryptedKey“

Referenz: [CommonPKI-3] Tabelle 8.#1
[RFC 5652], 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.

3.2.2.4 Datenfeld „ encryptedContentInfo“

Referenz: [RFC 5652], 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.

3.2.2.4.1 Teildatenfeld „contentType“

Referenz: [CommonPKI-3] Tabelle 9#1
[RFC 5652], 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.

3.2.2.4.2 Teildatenfeld „contentEncryptionAlgorithm“

Referenz: [CommonPKI-3] Tabelle 9#2
[RFC 5652], Abschnitt 6.1

Profilierung:
Als Verschlüsselungsalgorithmus für die Daten ist AES (id-aes256-CBC) vorgesehen; vgl. Abschnitt 2.1.3. Dabei ist unbedingt die Migrationsstrategie gemäß Abschnitt 1.1 zu beachten.

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

3.2.2.4.3 Teildatenfeld „encryptedcontent“

Referenz: [CommonPKI-3] Tabelle 9#3
[RFC 5652], Abschnitt 6.1

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

3.2.2.5 Datenfeld „unprotectedAttrs“ (entfällt!)

Referenz: [CommonPKI-3] 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 3.2.2.2

3.2.3 Transport der verschlüsselten Nachricht

Nachdem die signierte und verschlüsselte PKCS#7-Nachricht die Kombination eines Datenobjekt vom Typ SignedData und eines vom Typ EnvelopedData (vgl. 3.2„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>2

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

3.2.3.1 Transportformat

Referenz: [CommonPKI-3], Abschnitt 2.1.1,

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.

3.2.3.2 Transportwege

Die zu berücksichtigenden Transportwege zwischen den Kommunikationspartnern des ITSG-Trust Centers zum Austausch von Zertifizierungsanfragen sind in den Anlagen 7 (E-Mail) und 14 (Datenträger) spezifiziert.

3.3 Sichere Transportebene mit TLS (SSL)

Die Transport Layer Security (TLS), weitläufiger bekannt unter der Vorgängerbezeichnung Secure Socket Layer (SSL), ist ein kryptografisches Netzwerkprotokoll zur sicheren Datenübertragung im Internet. Das SSL-Protokoll wurde zuletzt 1996 in der Version 3.0 veröffentlicht, seitdem wird es unter dem neuen Namen TLS weiterentwickelt und standardisiert. Aktuell wird die TLS-Version 1.3 im RFC 8446 beschrieben.

[2] Die Dateinamen der verschlüsselten Nachrichten im Gesundheits- und Sozialwesen werden von der Sozialversicherung 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 keinen Suffix; vgl. Gemeinsame Grundsätze Technik

Für den elektronischen Datenaustausch im Gesundheits- und Sozialwesen sind die zu übermittelten Nutzdaten unabhängig von der Transportebene gemäß 3.1 zu signieren und zu verschlüsseln. Eine zusätzliche Transportsicherung mit dem TLS-Protokoll kann optional angewendet werden, um z.B. beim Verbindungsaufbau eine Authentisierung der Kommunikationspartner zu ermöglichen.

Bei der Nutzung einer Authentisierung beim Verbindungsaufbau und einer Transportverschlüsselung der zu übertragenen Daten ist mindestens die TLS-Version 1.2 zu nutzen.

3.3.1 Authentisierungsmethoden bei TLS-Verbindungen

In Abhängigkeit der Anforderung eines Dienstes zur Verwaltung und Authentisierung der Kommunikationspartner können für TLS-Verbindungen folgende Authentisierungsmethoden angewendet werden:

- einseitige Zertifikats-Authentisierung des Servers
- beidseitige Authentisierung von Server und Client
    - mit Serverzertifikat und Client-Login
    - beidseitige Zertifikats-Authentisierung von Server und Client

Die Authentisierung-Methode mit einseitiger Zertifikats-Authentisierung bietet keine ausreichende Sicherheit, da lediglich der Server authentisiert wird. Für eine sichere Authentisierung der Kommunikationspartner werden daher beidseitige Authentisierungs-Methoden empfohlen.

Für eine Zertifikats-Authentisierung gelten folgende Anforderungen an die Zertifikate:

- TLS-Zertifikate für Client und Server müssen von einer – selbst wiederum zertifizierten – Zertifizierungsstelle (trusted root) ausgestellt sein, die den Client und Server mit dessen Domain eindeutig identifiziert.
- Für spezielle Anwendungen im Gesundheits- und Sozialwesen gelten dabei für die Zertifikate folgende Anforderungen:
    - Zertifikate auf der Clientseite müssen von einer vertrauenswürdigen Zertifizierungsstelle im Gesundheits- und Sozialwesen ausgestellt sein; d.h. die vorhandenen Zertifikate aus den Arbeitgeber- und Leistungserbringerverfahren sind zu verwenden (siehe Kapitel 4.4).
    - Daraus ergibt sich, dass selbst-signierte Zertifikate (self-signed certificate), die ohne Beteiligung einer vertrauenswürdigen Zertifizierungsstelle erstellt wurden, nicht verwendet werden dürfen.

3.3.2 Vorgaben bei TLS-Verbindungen

Das hybride Netzwerkprotokoll TLS besteht aus einer Kombination von symmetrischen und asymmetrischen Algorithmen, um deren jeweilige Vorteile zu vereinen. Der Verbindungsaufbau erfolgt

im Public-Key-Verfahren und die nachfolgende Datenübermittlung wird mit symmetrischen Verfahren gesichert. Die diversen unterstützten Algorithmen sind in sogenannten Cipher-Suites spezifiziert, die beim Verbindungsaufbau zwischen Client und Server ausgehandelt werden.

Grundsätzlich gelten die Vorgaben in der technischen Richtlinie TR-02102-2 des BSI in der Version 2019-01.

4. Schlüsselmanagement

4.1 Allgemein

Die Integration eines asymmetrischen (Public-Key-Verfahren) Schlüsselmanagements begründet sich auf der Einrichtung einer vertrauenswürdigen Instanz (der Certification Authority - CA), der Schlüsselverwaltungsstelle oder dem Trust Center. Aufgabe des Trust Centers ist es, kryptographische Schlüssel sicher zu erzeugen, zu verwalten und zu verteilen. Das Schlüsselmanagement basiert auf der Verwendung von Zertifikaten zur Propagierung von öffentlichen Schlüsseln an Kommunikationspartner.

Das beschriebene Schlüsselmanagement umfasst insbesondere:

- die Zertifizierungshierarchie,
- die Zertifikate,
- die Erzeugung und Prüfung von Zertifikaten,
- das Sperrlistenmanagement sowie
- die Rollen innerhalb der Systemarchitektur.

Die öffentlichen Teilnehmerschlüssel werden durch eine vertrauenswürdige Instanz - der Certification Authority (CA) - zertifiziert. Dazu werden die öffentlichen Teilnehmerschlüssel in Form einer komplexen von der CA signierten Datenstruktur im System propagiert. Die Spezifikation der Zertifikate basiert auf dem aktuellen Zertifikatsformat X.509v3 (ITU-T X.509 97). Gegenüber dem bisherigen Zertifikatsformat sind zusätzliche Felder im Sinne von Zertifikatserweiterungen möglich.

Die Zertifikatsverifizierung durch den Leistungserbringer oder Arbeitgeber wird durch die Überprüfung der zugehörigen Gültigkeitsdauer und dem Abgleich einer durch die CA verteilten Sperrliste abgeschlossen. Die Identitäten von Subjekt und CA sind sogenannte „Distinguished Names", wie sie in X.500 festgelegt sind.

4.2 Namenskonventionen

Die unter X.500 vorzuhaltende Namenskonvention lautet:

C	=	Country	(DE)
O	=	Organization	(Name des Trust Centers)
OU	=	Organization Unit	(Name der Institution)

OU	=	Organization Unit	IK-Nummer oder Betriebs- bzw. Zahlstellennummer der Institution
CN	=	Common Name (Allgemeiner Name)	(Name des Ansprechpartners)

Die für die OU-Segmente gewählten Konventionen können im Rahmen der Pilotverfahren und der Interoperabilitätstests angepasst sowie um weitere OU-Segmente ergänzt werden.

Für die Zertifizierungsstellen lautet die unter X.500 vorzuhaltende Namenskonvention:

C	=	Country	(DE)
O	=	Organization	(Name des Trust Centers)

Als Zeichensatz zur einheitlichen Darstellungsform ist US-ASCII vorzusehen (Zeilenende CR/LF).

4.3 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).

4.4 Definition der Zertifikate

Die folgende Definition der Zertifikate enthält aufgrund des gewählten Ansatzes einer konzeptionellen Darstellung keine direkten Spezifikationen der Datenfelder. Zertifikate sind die zentralen Datenstrukturen innerhalb des Schlüsselmanagements für X.509. Dieser Abschnitt enthält einen Überblick über die Inhalte eines Zertifikats. Im Anhang 6.1 befindet sich die ASN.1 Syntax eines X.509v3-Zertifikats. Ein Zertifikat besteht danach aus den folgenden Datenfeldern:

Referenz: [CommonPKI-1], 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 Interoperablititätsanforderungen.

4.4.1 Versionsnummer

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 Integer Werte „0“, „1“ oder „2“ sind entsprechend angegeben.

Profilierung:
Es sind ausschließlich X.509v3-Zertifikate zu verwenden, der Integer Wert für das Datenfeld version beträgt „2“.

4.4.2 Seriennummer

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.

Beispiel3: 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 ... 2^{159} zur Verfügung, das verbleibende Bit (Most Significant Bit) ist zur Darstellung des positiven Vorzeichens zu verwenden. Führende Nullen (Ziffer) sind gestattet.

4.4.3 Signaturalgorithmus

Dieses Datenfeld informiert über den Algorithmus und die zugehörigen Parameter, die vom Zertifikatserzeuger zur Signaturbildung verwendet werden.

Referenz: [CommonPKI-1], Kapitel 2, Tabelle 2.#3

Das Signatur-Datenfeld signature dient der Bezeichnung des Algorithmus, mit dem die Signatur gebildet wird4. Die Struktur ist mit der des Feldes signatureAlgorithm identisch (s. Abschnitt 4.4.11). 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:
Die Inhalte der Datenfelder signatureAlgorithm (aus der Datenstruktur Certificate) und signature (aus der Datenstruktur „tbsCertificate“) müssen identisch sein. Die Security Schnittstelle legt fest, dass RSASSA-PSS einzusetzen ist; vgl. Abschnitt 2.1.2.

---

[3] 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.

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

4.4.4 Name der Zertifizierungsstelle

Referenz: [CommonPKI-1], 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 gemäß [CommonPKI-1] kein Gebrauch gemacht werden. Die Menge der AttributeTypeAndValue-Komponenten eines RelativeDistinguishedName darf nur ein Element enthalten.

[CommonPKI-1] 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 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. - „ITSG TrustCenter fuer Arbeitgeber" - „ITSG TrustCenter fuer sonstige Leistungserbringer" - „DKTIG TrustCenter fuer Krankenhaeuser und Leistungserbringer PKC"

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
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 UTF8String5, 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 nicht gefordert, obwohl ISIS-MTT ("must") dieses für alle Zertifikate fordert.

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

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

Für die Werteangaben dürfen dabei lediglich die folgenden Zeichen verwendet werden:
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789/ -.()

4.4.5 Name von Zertifikatsinhabern

Referenz: [CommonPKI-1], 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.

[5] UTF steht für Universal Character Set (UCS) Transformation Format. UTF-8 ist eine 8-Bit-Kodierung variabler Länge (s. [RFC 3629 2003]).

Das subject-Feld hat dasselbe Format wie das oben bereits beschriebene issuer-Feld. Auch für das subject-Feld sind nur X.500-Distinguished-Names zulässig.

Wie oben beschrieben, lassen sich im vorliegenden Profil die beiden Typen von DNs "allgemeine Teilnehmerzertifikate" und „Zertifizierungsstellen (PCA oder CA)" unterscheiden.

Für das Feld subject ist der Erstere 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	Organization-Name (verpflichtend, fest)	O	Name des Trust Centers als feste Zeichenkette - „ITSG TrustCenter fuer Arbeitgeber" - „ITSG TrustCenter fuer sonstige Leistungserbringer" - „DKTIG TrustCenter fuer Krankenhaeuser und Leistungserbringer PKC"
3	Organization-UnitName (verpflichtend)	OU	Name der Institution (Firmenname des Leistungserbringers oder des Arbeitgebers)
4	Organization-UnitName (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 DNs 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 4.4.4.

Die möglichen Werte für ein Attribut werden durch die Typangabe gemäß Abschnitt 4.4.4 bestimmt.

Profilierung:
Die oben in der Tabelle ausgeführten Angaben zum DN sind verbindlich. Im Übrigen gelten die Anforderungen an DNs 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 [CommonPKI-1] vorgesehene Pseudonyme-Regelung nach dem 31.12.2003 wird nicht unterstützt, da eine Pseudonymisierung des Zertifikatsinhabers nicht wünschenswert ist.

4.4.6 Gültigkeitsdauer

Referenz: [CommonPKI-1], 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 }
```

Profilierung:
Bei der Generierung von Zertifikaten ist UTCTime zu verwenden, d. h. die Verwendung von GeneralizedTime ist verboten[6].
Die Datums- und Zeitangaben müssen im Format YYMMDDHHMMSSZ[7] erfolgen, das in folgender Tabelle beschrieben wird:

[6] Anmerkung: Nach dem Profil des NIST ist stets „UTCTime" zu verwenden. Das SigI-Profil sieht stets „GeneralizedTime" vor. Die IETF schreibt vor, ab dem Jahr 2050 „GeneralizedTime" zu verwenden, davor „UTCTime". Entsprechend dem Ansatz des SigI-Profils soll so schnell wie möglich auf „GeneralizedTime" umgestellt werden.

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

Datumsangaben		Zeitangaben	
Feld	**Bedeutung**	**Feld**	**Bedeutung**
YY	zweistellige Jahreszahl	HH	Stunde: 00, 01, ..., 23
MM	Monat 01, 02, ..., 12	MM	Minute 00, 01, ..., 59
DD	Tag 01, 02, ..., 31	SS[8]	Sekunde 00, 01, ..., 59

4.4.7 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:

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

4.4.8 Öffentlicher Schlüssel des Zertifikatsinhabers

Referenz: [CommonPKI-1], Kapitel 2, Tabelle 2.#8 und 2.#14

Das Feld subjectPublicKeyInfo ist eine Datenstruktur, die den öffentlichen Schlüssel des Zertifikatsinhabers im Teilfeld subjectPublicKey und die zulässigen Algorithmen, mit denen die Verwendung des Schlüssels zulässig ist, im Teilfeld algorithm enthält.

Es wird folgende Datenstruktur verwendet:

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

Die Belegung des Feldes AlgorithmIdentifier erfolgt gemäß Abschnitt 2.1.5.

In Kapitel 2 werden außerdem 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.

4.4.9 Syntax für x.509v3 Zertifikate

Der X.509v3 Standard beinhaltet eine Vielzahl an Zertifikatserweiterungen. Die Zertifikate und Sperrlisten müssen die für die Prüfung der Gültigkeit (einschließlich der elektronischen Signatur)

[8] Sekunden sind stets anzugeben, auch wenn die Anzahl der Sekunden Null ist.

notwendigen Informationen enthalten. Soweit daneben Extensions verwendet werden, ist eine vorherige Abstimmung zwischen allen am Verfahren beteiligten erforderlich. Als Basis gelten die Festlegungen der Profile für Zertifikate und Sperrlisten nach den entsprechenden MTTv2 -Spezifikationen.

4.4.10 Signatur der CA

Referenz: [CommonPKI-1], 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 wird[9].

4.4.11 Identifizierung des verwendeten Signaturalgorithmus

Referenz: [CommonPKI-1], 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 }
```

Im Feld algorithm wird ein Algorithmus angegeben. 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.
In Kapitel 2 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 RSAES-PSS einzusetzen ist; vgl. Abschnitt 2.1.2 Signaturalgorithmen. Die Datenfelder gemäß dem verwendeten Algorithmus zu belegen (vgl. Abschnitt 2.1.2.1).

4.4.12 Nicht verwendete optionale Datenfelder

Eindeutige Bezeichner für CAs und Zertifikatsinhaber

[9] 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.

Referenz: [CommonPKI-1], Kapitel 2, Tabelle 2.#8 und 2.#14

Profilierung:
Gemäß [CommonPKI-1, 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 4.4.4) und subject (vgl. Abschnitt 4.4.5) eingetragenen DNs eindeutig sind.

4.5 Gültigkeitszeitraum der Zertifikate

Die Teilnehmer-Zertifikate haben grundsätzlich eine Gültigkeitsdauer von maximal drei Jahren. Einzelne Trust Center bzw. die einzelne CA (Certification Authority) können im Rahmen ihrer Dienstleistung einen kürzeren Zeitraum für die Gültigkeit ihrer Teilnehmer-Zertifikate vorsehen. Für die PCA (Policy Certification Authority) beträgt die Gültigkeitsdauer des Zertifikates 7 Jahre und für die CA 5 Jahre.
Die Security Schnittstelle sieht die Anwendung eines Schalenmodells für die Zertifikatshierarchie vor. Voraussetzung für Anwendungen, die das Schalenmodell unterstützen ist dabei, dass die Laufzeit eines Zertifikats vollständig innerhalb der Laufzeit des ausstellenden (CA-) Zertifikates liegt. Dies wird dann zu einem Problem, wenn die CA-Zertifikate nicht mehr lange gültig sind, da man dann nur noch für den verbleibenden Zeitraum Zertifikate ausstellen kann. Um dieses Problem zu beheben, hat das Zertifikat der PCA eine Laufzeit von 7 Jahren. Bereits nach fünf Jahren wird das nächste Zertifikat erzeugt, mit dem von da an zertifiziert wird (das alte Zertifikat stellt dann nur noch die Gültigkeit der vorher ausgestellten Zertifikate, insbesondere durch die regelmäßige Erzeugung aktueller Sperrlisten, sicher). Somit ist es für die PCA jederzeit möglich, CA-Zertifikate mit einer Laufzeit von 5 Jahren zu erzeugen.
Die beteiligten Trust Center sind verantwortlich für die Einhaltung dieser Konvention. Sie prüfen vor der jeweiligen Teilnehmerzertifizierung die Laufzeiten des eigenen CA-Zertifikats und des PCA-Zertifikats und stellen sicher, dass die Laufzeiten nicht überschritten werden. Dazu beantragen die Trust Center rechtzeitig vor dem Auslauf des CA-Zertifikates die Neu-Zertifizierung durch die PCA.

4.6 Öffentliche Schlüsselverzeichnisse

Alle öffentlichen Schlüssel der Teilnehmer des Datenaustauschverfahrens im Gesundheits- und Sozialwesen werden in öffentlichen Schlüsselverzeichnissen veröffentlicht, damit eine Überprüfung möglich ist, welche zertifizierte Schlüssel für das Verfahren registriert sind. Die Empfänger der Daten (z.B. Datenannahmestellen) können anhand der öffentlichen Schlüssel-Verzeichnisse die Absender einer verschlüsselten Nachricht eindeutig identifizieren. Die öffentlichen Schlüssel werden

in Dateiform oder als LDAP-Verzeichnis zur Verfügung gestellt. Die Schlüsselaktualisierung erfolgt an Werktagen von Montag bis Freitag.

4.6.1 Schlüssellisten

Die Schlüssellisten im Dateiformat enthalten öffentliche Schlüssel in base64-Kodierung. Sie enthalten die gültigen Schlüssel der beiden Trust Center der DKTIG und der ITSG und werden von der ITSG zum Download zur Verfügung gestellt. In Abhängigkeit der öffentlichen Teilnehmerschlüssel in den einzelnen Schlüssellisten sind die zugeordneten PCA- und der CA-Schlüssel enthalten, wobei alle PCA- und CA-Schlüssel nur in den Gesamtlisten mit allen öffentlichen Schlüsseln enthalten sind. Diese haben im Rahmen des Schalenmodells einen überlappenden Gültigkeitszeitraum (siehe Kapitel 4.5).
Zusätzlich stellt jedes Trustcenter tagesaktuell eine Liste der AG-Meldestellen und eine Liste der LE-Meldestellen den SV-Trägern zur Verfügung. Diese Liste besteht aus den Betriebsnummern bzw. Institutionskennzeichen aller Teilnehmer, die über ein gültiges Zertifikat mit dem Attribut „AG-Meldestelle“ bzw. „LE-Meldestelle“ verfügen.

4.6.1.1 Übersicht der verfügbaren öffentlichen Schlüssellisten

Für das Arbeitgeberverfahren existieren folgende Schlüssellisten mit öffentlichen Teilnehmerschlüsseln:

- gesamt-pkcs.agv (alle Teilnehmerschlüssel mit 2048 und 4096 Bit Schlüssellänge)
- gesamt-sha256.agv (optional, alle Teilnehmerschlüssel mit 2048 Bit Schlüssellänge)
- gesamt-rsa4096.agv (optional, alle Teilnehmerschlüssel mit 4096 Bit Schlüssellänge)
- annahme-sha256.agv (Schlüssel der Datenannahmestellen mit 2048 Bit Schlüssellänge)
- annahme-rsa4096.agv (Schlüssel der Datenannahmestellen mit 4096 Bit Schlüssellänge)
- sperrliste-ag-sha256.crl (gesperrte Teilnehmerschlüssel mit 2048 Bit Schlüssellänge)
- sperrliste-ag-rsa4096.crl (gesperrte Teilnehmerschlüssel mit 4096 Bit Schlüssellänge)

Für das Leistungserbringerverfahren existieren folgende Schlüssellisten mit öffentlichen Teilnehmerschlüsseln:

- gesamt-pkcs.key (alle Teilnehmerschlüssel mit 2048 und 4096 Bit Schlüssellänge)
- gesamt-sha256.key (optional, alle Teilnehmerschlüssel mit 2048 Bit Schlüssellänge)
- gesamt-rsa4096.key (optional, alle Teilnehmerschlüssel mit 4096 Bit Schlüssellänge)
- annahme-sha256.key (Schlüssel der Datenannahmestellen mit 2048 Bit Schlüssellänge)
- annahme-rsa4096.key (Schlüssel der Datenannahmestellen mit 4096 Bit Schlüssellänge)
- pkv-sha256.key (Sonderliste mit Schlüssel der PKV mit 2048 Bit Schlüssellänge)
- pkv-rsa4096.key (Sonderliste mit Schlüssel der PKV mit 4096 Bit Schlüssellänge)
- sperrliste-le-sha256.crl (gesperrte Teilnehmerschlüssel mit 2048 Bit Schlüssellänge)
- sperrliste-le-rsa4096.crl (gesperrte Teilnehmerschlüssel mit 4096 Bit Schlüssellänge)

Nach Abschluss der Migration Anfang 2023 werden alle Schlüssellisten mit „sha256“ hinfällig und nicht weiter zur Verfügung gestellt.

4.6.2 LDAP-Verzeichnis

In diesem Abschnitt werden die LDAP-Verzeichnisdienste beschrieben, die gemäß Abschnitt 1.4 zur Anwendung kommen. Der Verzeichnisdienst basiert auf der aktuellen Version LDAP v3, die im [RFC 4511] dokumentiert ist und berücksichtigt die Spezifikationen von [CommonPKI-4]. Das LDAP-Verzeichnis wird von der ITSG ausschließlich für Datenannahmestellen bereitgestellt.

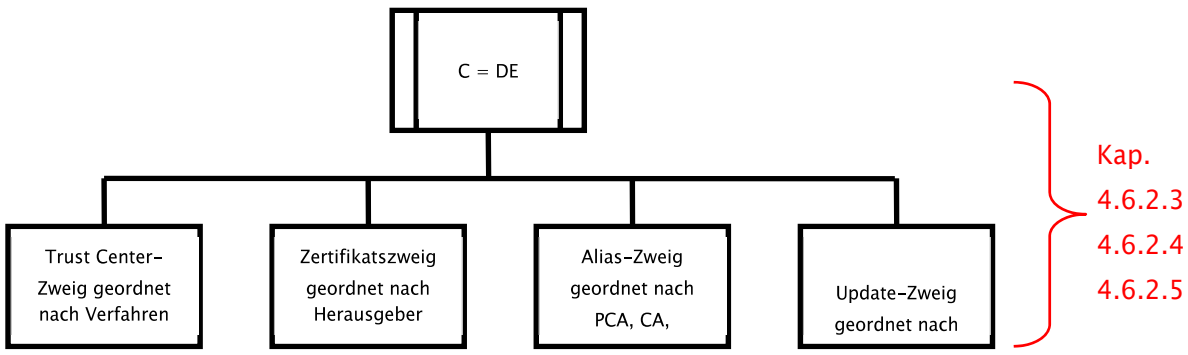
4.6.2.1 Aufbau des LDAP-Verzeichnis

Referenz: Abschnitt 2.3 [RFC 4511]
[RFC 4513]
[CommonPKI-4]

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

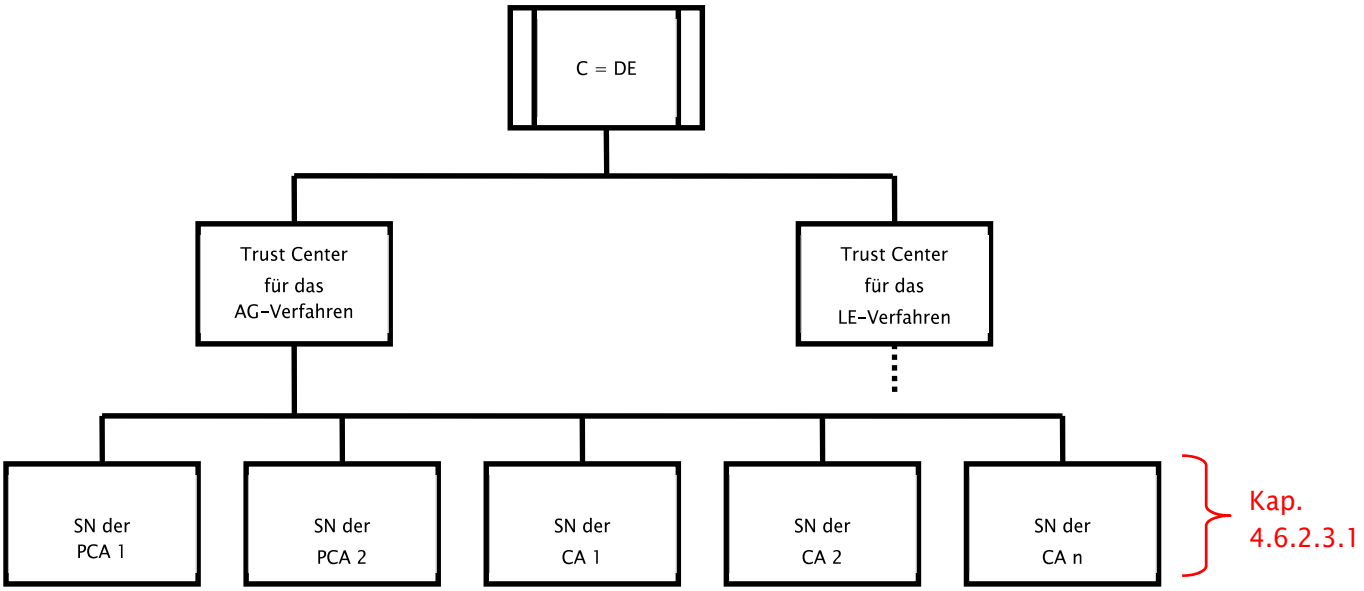
Der LDAP-Server wird über den Standard-Port 389 (LDAP) betrieben.

4.6.2.2 Aufbau der Wurzel

Name	Object Class	Attribute
De	Country	• countryName=de

4.6.2.3 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.



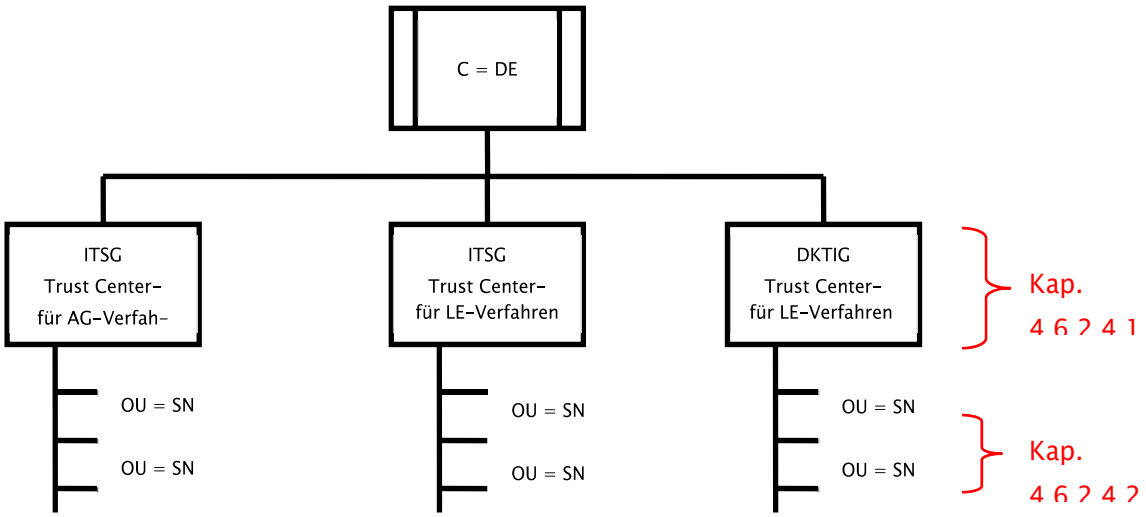
4.6.2.3.1 Aufbau der PCA- und CA-Ebene

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

Name	Object Class	Attribute
<Seriennummer aus dem Zertifikat>	Organization	• Trust Center-Name • Seriennummer des Zertifikats • Zertifikat binär

4.6.2.4 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.



4.6.2.4.1 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.

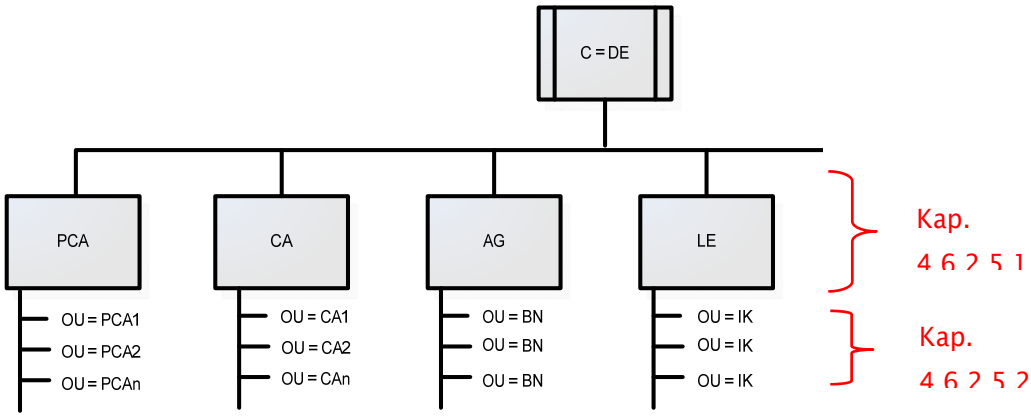
Name	Object Class	Attribute
<OrganizationName aus dem Zertifikat>	Organization Unit	certificateRevocationList = <CRL der CA> Seriennummer des Zertifikats

4.6.2.4.2 Aufbau der Zertifikats-Ebene

Name	Object Class	Attribute
<Seriennum-mer des Zertifi-kats>	Organization Unit	• Firmenname • Institutionskennzeichen oder Betriebs- bzw. Zahlstellennummer • Seriennummer des Zertifikats • Zertifikat (binär)

4.6.2.5 Aufbau des Alias-Zweigs

Die nachfolgende Abbildung zeigt den Directory Information Tree für den suchoptimierten Alias-Zweig geordnet nach PCA, CA, AG und LE. 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.



4.6.2.5.1 Aufbau der PCA-, CA -, AG-, LE- Ebene

Für die PCA, CA, AG und LE wird ein eigenes Objekt der Klasse „Organization“ angelegt. In einer Übergangsphase werden ehemalige Zweige für Annahmestellen AG und LE aus dem LDAP-Verzeichnis entfernt.

Name	Object Class	Attribute
<Organization-Name aus dem Zertifikat>	Organization	• Trust Center-Name oder • Institutionskennzeichen oder Betriebs- bzw. Zahlstellennummer

4.6.2.5.2 Aufbau der Verweis-Ebene

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

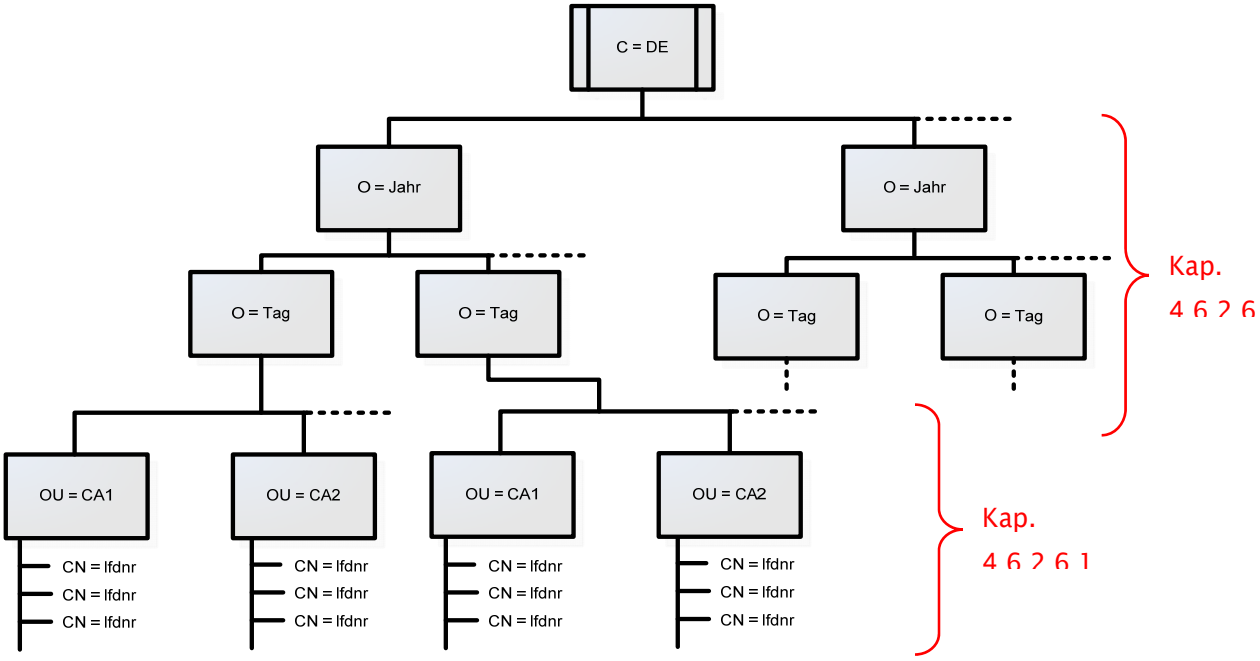
Name	Object Class	Attribute
<Seriennummer des Zertifikats>	Organization Unit	• Trust Center-Name oder Firmenname • Seriennummer des Zertifikats • Referenz zum Zertifikat (seeAlso) • Gültigkeit des Zertifikats „von“ • Gültigkeit des Zertifikats „bis“ • Verschlüsselungsverfahren • Ansprechpartner im Fall AG oder LE

4.6.2.6 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. Um den Datenbestand für eine Daten-Replizierung in einem notwendigen Rahmen zu halten, werden in den Update-Zweigen nur gültige Schlüssel gespeichert. Ungültige Schlüssel werden regelmäßig aus den Update-Zweigen gelöscht. Das Löschen von Daten in den Update-Zeigen wird in einer Übergangsphase umgesetzt.



4.6.2.6.1 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	• Trust Center-Name

4.6.2.7 Zugriffsrechte und -rollen

Die Annahmestellen der Sozialversicherung 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 kann über LDAP (Port 389) zugegriffen werden.

4.6.2.8 Suchfunktionen

Die Suche nach Binärdateien der Teilnehmer-Zertifikate ist über das Institutionskennzeichen oder über die Betriebs- bzw. Zahlstellennummer 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.

4.6.2.9 Zertifikatshandling

4.6.2.9.1 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.

4.6.2.9.2 Abgelaufene Zertifikate

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

4.6.2.9.3 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. Zahlstellennummer 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. Zahlstellennummer entweder im Zertifikatszweig im Aliaszweig AG oder LE 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.

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

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

4.6.2.9.4 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 4.6.2.4.1) 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.

4.6.2.10 Betrieb des LDAP-Dienstes

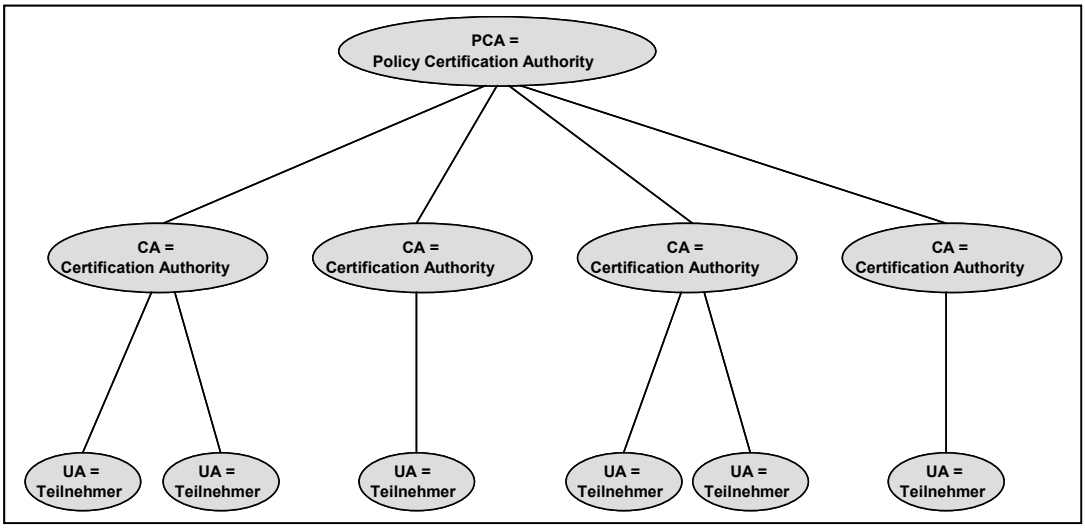
Jedes Trust Center, das am Datenaustausch-Verfahren teilnimmt, stellt einen eigenen LDAP Dienst zu Verfügung der von dem jeweiligen Trust Center eigenständig betrieben und gepflegt wird.

Die Trust Center können untereinander eine automatisierte Replizierung vereinbaren, oder Sie können in bilateraler Absprache alle öffentlichen Schlüssel über die ITSG zum Abruf bereitstellen.

5. PKI-Verfahrensbeschreibung

5.1 Struktur der Zertifizierungshierarchie

Die Zertifizierungshierarchie folgt nachfolgendem Aufbau.



Die oberste Wurzel des Zertifizierungsbaumes ist die Policy Certification Authority. Auf der nächsten Ebene der Zertifizierungshierarchie befinden sich die CA'en um Teilnehmer und weitere Strukturen einer Organisation zu zertifizieren. Zurzeit wird die PCA gebildet durch die Kooperationsgemeinschaft Trust Center unter Leitung der Informationstechnischen Servicestelle der gesetzlichen Krankenversicherung (ITSG GmbH). Der Vertrag zur Kooperation regelt den Zutritt weiterer Trust Center, sofern die Bedingungen der Policy der PCA erfüllt werden.

5.2 Rollen und ihre Funktionen

Die Systemarchitektur wird anhand der im System vorhandenen Rollen und ihren Funktionalitäten beschrieben. Innerhalb der Architektur können danach die folgenden vier Rollen identifiziert werden:

- Teilnehmer (UA),
- Zertifizierungsstelle (CA),
- Registrierungsstelle (RA) und
- Verzeichnisdienst (DIR).

5.3 PCA-Wurzel der Zertifizierungshierarchie

Die PCA (Policy Certification Authority) verfügt über eine Policy, in dem die PCA-Sicherheitspolitik festgelegt wird. Eine signierte Kopie dieses Dokumentes wird allgemein verfügbar gemacht. Die Autorisierung einer CA, innerhalb der Zertifizierungshierarchie zu operieren, basiert auf diesem Dokument sowie auf dem herausgegebenen Zertifikat der PCA.

Die Sicherheitspolitik folgt dabei den Sicherheitsleitlinien einer obersten Zertifizierungsstelle, die allgemein als Policy Certification Authority (PCA) bezeichnet wird. Es haben sich die von den Spitzenverbänden der gesetzlichen Krankenkassen eingerichtete

Informationstechnische Servicestelle der

Gesetzlichen Krankenversicherung GmbH (ITSG)

die von der Deutsche Krankenhausgesellschaft eingerichtete

Deutsche Krankenhaus TrustCenter und
Informationsverarbeitung GmbH (DKTIG)

und der

Datenstelle der Rentenversicherung (DSRV),
unterhalten von der Deutschen Rentenversicherung Bund

auf die gemeinsame Gestaltung der PCA Datenübermittlung im Gesundheits- und Sozialwesen und deren Policy verständigt.

5.3.1 Identität der PCA

Die o. g. Organisationen betreiben die PCA als gleichberechtigte Partner. Mit dem Aufbau und der Wahrnehmung der DV-technischen Aufgaben der PCA ist derzeit die Atos Information Technology GmbH betraut.

5.3.2 Zuständigkeitsbereich der PCA

Als Zuständigkeitsbereich der PCA ist vorrangig das deutsche Gesundheits- und Sozialwesen vorgesehen. Das primäre Ziel besteht in der Sicherung der Kommunikation im Rahmen des Datenaustausches zwischen der GKV, GRV und allen Leistungserbringern, die über eine IK-Nummer, sowie Arbeitgebern/Zahlstellen, die über eine Betriebsnummer oder Zahlstellennummer verfügen.

Der Name der PCA wurde daher so gewählt, dass nicht nur CA's aus dem Gesundheitswesen oder dem Bereich der Rentenversicherung, sondern darüber hinaus ggf. auch CA's und Teilnehmer aus anderen Bereichen des Sozialwesens im Interesse einer für alle Beteiligten vereinfachten Gestaltung der CA (insbesondere auch im Hinblick auf den Austausch der von ihnen zertifizierten Schlüssel) die Funktion der PCA in Anspruch nehmen können.

PCAs sollten substantiell unterschiedliche Sicherheitsleitlinien haben. Für das deutsche Gesundheits- und Sozialwesen sind aus Sicht der genannten Partner substantiell unterschiedliche Sicherheitsleitlinien nicht erforderlich. Darüber hinaus sind die vorhandenen Sicherheitsleitlinien auch für weitere Bereiche des Sozialwesens ausreichend (z. B. für Arbeitgeber für die Übermittlung von DEÜV-Meldungen und Beitragsnachweise).

5.4 Trust Center (Certification Authority)

5.4.1 Zertifizierungsanforderung

Die CA (Trust Center) generiert auf Anfrage ein Zertifikat das u.a. den Namen des Systemteilnehmers, den öffentlichen Schlüssel sowie den Namen des Zertifikatserzeugers enthält. Dabei signiert

die CA diese Daten unter Verwendung ihres privaten Schlüssels. Für diesen Vorgang sind vom Teilnehmer bzw. UA mindestens der öffentliche Schlüssel und der Name mitzuteilen. Innerhalb dieses Prozesses ist es erforderlich, dass die CA den Teilnehmer authentisiert, d. h. sich von der Korrektheit der Teilnehmeridentität überzeugt, bevor sie die Zusammengehörigkeit von Teilnehmername und öffentlichem Schlüssel durch das zu erzeugende Zertifikat bestätigt (siehe RA). Die dazu eingesetzten Mechanismen werden durch die CA festgelegt.

Zur Zertifikatsanforderung (Certification Request) durch einen Teilnehmer an eine CA werden im Folgenden die Varianten

- Anforderung mittels PKCS#10-Request per E-Mail,
- Anforderung mittels PKCS#10-Request über eine Online-Schnittstelle (nur ITSG-Trust Center),
- Anforderung mittels PKCS#10-Request via Datenträger (z. B. Diskette, DVD)

beschrieben.

Im ersten Fall erfolgt die Kommunikation zwischen dem Teilnehmer und dem Trust Center mittels E-Mail. Optional können Trust Center weitere Kommunikationswege unterstützen. Sie unterscheiden sich durch die von der CA vorgenommenen Form der Authentikation der anfordernden Teilnehmer.

5.4.2 Zertifikatsüberprüfung

Ein erster Schritt einer jeden Zertifikatsprüfung ist die Verifizierung der Signatur des Zertifikatserzeugers unter Verwendung des zugehörigen öffentlichen Schlüssels.

Die Prüfung umfasst folgende Schritte:

1. Erfolgreiche Validierung des Certification Requests (PKCS#10),
2. erfolgreiche Verifizierung der Signatur bzw. des Hashwertes des öffentlichen Schlüssels,
3. sowie einer positiven Konsistenzprüfung beider Nachrichteninhalte.
4. Der Teilnehmer bzw. Antragsteller erhält eine entsprechende Nachricht, sofern die Prüfung ein negatives Ergebnis erbringt.

5.4.3 Eindeutigkeit von Namen

Eine wesentliche Anforderung an das Zertifizierungsschema ist die Eigenschaft der Namenseindeutigkeit aller Knoten und insbesondere aller zertifizierenden Trust Center.

Bei Zertifizierung einer CA richtet die zertifizierende PCA / CA eine Anfrage an die Datenbank. Besteht kein Konflikt bezüglich dieses Datums, so kann die CA in der Datenbank registriert werden. Dazu sind von der PCA / CA Name, öffentlicher Schlüssel und Name der CA anzugeben.

5.4.4 Propagierung Zertifizierungsinformation

Jeder Sender einer signierten Nachricht muss dem Empfänger die notwendige Zertifizierungsinformation zur Verfügung stellen, d. h. im Zweifelsfalle die vollständige Information, um alle Zertifikate des Zertifizierungspfads verifizieren zu können.

Steht ein geeignetes Public Directory zur Verfügung, so kann auf die Übermittlung der Zertifizierungsinformation verzichtet werden, falls dem Empfänger der Nachricht die Nutzung des Directorys möglich ist.

5.4.5 Sperrlisten Management

Anwendungen für einen rechtsverbindlichen Geschäftsverkehr mit Einsatz der elektronischen Signatur erfordern einen Verzeichnisdienst auch zum Abruf von Sperrlisten (alternativ können CAs auf Online-Sperrbenachrichtigungsmechanismen wie z. B. das OCSP-Protokoll zurückgreifen).

Jede CA ist verantwortlich für die Ausgabe der von ihr gesperrten Zertifikate. Für die Sperrung eines Zertifikates können mehrere Gründe ausschlaggebend sein:

- das Schlüsselpaar wurde kompromittiert oder es besteht ein begründeter Verdacht der Kompromittierung und
- organisatorische Gründe (z.B. die Entfernung des Teilnehmernamens aus dem System),
- falsche Angaben im Zertifikat.

In Anhang 6.1.3 „Sperrliste“ ist die ASN.1 Syntax einer Sperrliste wiedergegeben. Es ist das Format „CRLv2“ nach der Spezifikation X.509v3 (ITU-X.509 97) zu unterstützen.

Eine Sperrliste besteht aus den folgenden Einträgen:

- **Signatur**
  (Identität des Signaturalgorithmus und zugehörige Parameter)

Dieses Datenelement entspricht dem gleichnamigen Datenfeld im Zertifikat.

- **Erzeuger der Sperrliste**
  Name der CA, die die Sperrliste signiert hat.
  Ausgabedatum
  Datum der Listenerstellung.
  nächste Aktualisierung
  Hier wird der vorgesehene Zeitpunkt zur Verteilung der nächsten aktualisierten Sperrliste angegeben.
  Liste der gesperrten Zertifikate

Für jedes gesperrte Zertifikat wird die zugehörige Seriennummer und der Zeitpunkt der Sperrung angegeben.

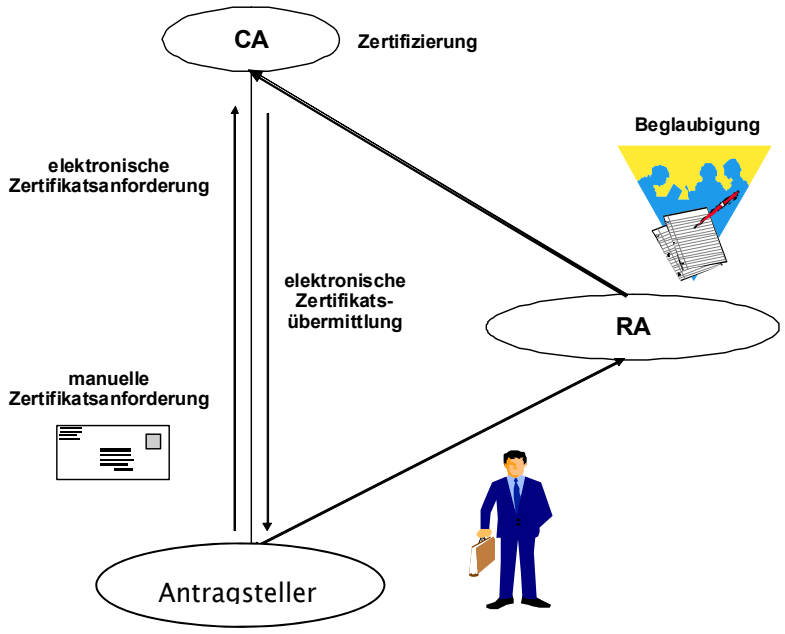
Die CRL's werden sofort nach einer Sperrung, spätestens alle 2 Wochen aktualisiert.

5.5 Registrierungsstelle (RA = Registration Authority)

Eine Identifizierung und Authentifizierung des Teilnehmers kann durch eine Registrierungsstelle unterstützt werden, die auch räumlich getrennt von der CA agieren kann. Die Beglaubigung des Antrages wird von der RA an die CA übermittelt. In diesem Fall wird die RA a priori als vertrauensvolle Instanz angesehen.

Die Zertifikatsübermittlung durch die CA an den Teilnehmer erfolgt in der beschriebenen elektronischer Form.

Die beschriebenen Abläufe für eine Certification Request (Zertifikatsanforderung) sind im nachfolgend veranschaulicht:



Die CA nimmt die Zertifizierung der Schlüssel vor, muss sich dabei jedoch in geeigneter Form davon überzeugen, dass der Antragsteller tatsächlich derjenige ist, der er zu sein vorgibt.

Zur Beglaubigung des Antrages sind vom Antragsteller neben dem ausgefüllten Antrag auf Zertifizierung, der postalisch oder elektronisch gestellt werden darf, die nachstehend folgenden Schritte umzusetzen:

- Zusendung eines unterschriebenen Ausdrucks eines eindeutigen Hashwerts seines öffentlichen Schlüssels (postalisch)

- Nachweis der Identität durch eine geeignete Maßnahme, die die Anforderungen an eine substantielle Authentifizierung des BSI erfüllt (z.B. PostIdent oder eID-Funktion des Personalausweises)
- im AG-Verfahren bei einem Erstantrag Angabe der Betriebsnummer. Das Trust Center muss durch geeignete Maßnahmen prüfen, ob die Betriebsnummer dem Antragsteller zugeordnet werden darf. Dies kann durch einen Freischaltcode erfolgen, der postalisch an die in der Betriebsstättendatei der Bundesagentur für Arbeit hinterlegte Adresse versendet wird. Dieser Freischaltcode muss vom Antragsteller mit dem Schlüsselpaar verknüpft ans Trust Center gesendet werden (idealerweise durch eine Signatur des Freischaltcodes mit dem privaten Schlüssel aus dem Antrag).
- im LE-Verfahren bei einem Erstantrag Angabe des IK. Das Trust Center muss durch geeignete Maßnahmen prüfen, ob das IK dem Antragsteller zugeordnet werden darf. Dies kann durch einen Freischaltcode erfolgen, der postalisch an die in dem Verzeichnis der ArGe IK hinterlegte Adresse versendet wird. Dieser Freischaltcode muss vom Antragsteller mit dem Schlüsselpaar verknüpft ans Trust Center gesendet werden (idealerweise durch eine Signatur des Freischaltcodes mit dem privaten Schlüssel aus dem Antrag).

Der Registrierungsstelle ist es gestattet, dem bereits identifizierten Antragsteller die hinterlegten Adressen zu seiner Betriebsnummer / seinem IK anzuzeigen, damit dieser (bei mehreren hinterlegten Adressen) die korrekte auswählen kann. Weiter kann der Antragsteller dadurch reagieren, falls die Adressdaten nicht mehr aktuell sind. Die Registrierungsstelle darf den Freischaltcode an keine alternative Lieferadresse außerhalb der Registerdaten versenden, der Antragsteller muss beim zuständigen Betreiber des Registers eine Änderung initiieren.
Wenn ein Antragsteller für weitere Betriebsnummern oder Institutionskennzeichen Meldungen abgeben möchte, muss er entweder für jede Betriebsnummer/jedes IK ein eigenes Zertifikat beantragen oder ein Zertifikat als Arbeitgeber-Meldestelle bzw. Leistungserbringer-Meldestelle beantragen. Diesem Antrag muss zusätzlich eine Eigenerklärung beigefügt werden, dass ausschließlich Meldungen für Meldepflichtige getätigt werden, die gegenüber der AG-/LE-Meldestelle oder deren Vertragspartner nach substantiellem Sicherheitsniveau identifiziert wurden. Die Identifikation des Vertragspartners durch die AG-/LE-Meldestelle muss dabei nach substantiellen Sicherheitsniveau erfolgt sein. Muster für die Eigenerklärung sind dieser Anlage als Anhang beigefügt. Für die AG-/LE-Meldestellen wird im Trust Center eine Zuordnungstabelle gepflegt, welche Betriebsnummern/Institutionskennzeichen sich erfolgreich als Meldestelle registriert haben.
Die Listen der Betriebsnummern bzw. der Institutionskennzeichen aller AG-/LE-Meldestellen werden tagesaktuell an alle Sozialversicherungsträger und deren Dienstleister verteilt, damit beim Dateneingang Prüfungen durchgeführt werden können, ob der Absender befugt ist, für andere zu melden.

Eine Sonderstellung nehmen die Sozialversicherungsträger und deren Dienstleister inkl. der Datenannahmestellen ein. Da diese in einem gemeinschaftlichen Prozess zu einem bestimmten Stichtag ihre Zertifikate wechseln, wird hier ein separater manueller Prozess durchgeführt, bei dem manuell die Echtheit und Authentizität des Antragsstellers festgestellt wird.

Bei einem papierlosen Online-Folgeantrag sind keine Antragsunterlagen beim ITSG-Trust Center einzureichen. Der Online-Folgeantrag muss mit einem gültigen Zertifikat vom ITSG-Trust Center +elektronisch signiert werden. Online-Folgeanträge dürfen nur durch den identifizierten Antragsteller gestellt werden. Ansonsten ist ein neuer Erstantrag erforderlich.
AG-/LE-Meldestellen dürfen den papierlosen Online-Folgeantrag für einen Zeitraum von maximal 3 Jahren nach erfolgreicher Registrierung nutzen. Nach diesem Zeitraum ist ein erneuter Erstantrag erforderlich.

5.5.1 Übergangsszenario

Ab dem 1. April 2021 wird die geänderte Authentifizierung und Identifizierung umgesetzt. Die Prüfung der Meldestellen erfolgt zeitversetzt ab dem 1. Juli 2023. Sofern eine Meldestelle bis dahin kein neues Zertifikat beantragen muss, ist die ggf. notwendige Eigenerklärung bis zum 31. März 2023 gegenüber dem zuständigen Trustcenter abzugeben.

5.6 Teilnehmer

Die Teilnehmer nehmen unter Verwendung technischer Hilfsmittel (Hard- und Software) als Initiatoren (Sender) bzw. eigentliche Endabnehmer (Empfänger) am System teil. Dabei kann der im Auftrag eines Teilnehmers agierende Prozess als Teilnehmer-Repräsentant angesehen werden. Diese Sichtweise wird durch den separaten Begriff "User Agent" (UA) unterstützt. Teilnehmer bzw. UA müssen nachfolgende Funktionalitäten zur Verfügung stellen.

5.7 Erzeugung und Schutz der Teilnehmerschlüssel

Die Schlüsselpaare sind vom Teilnehmer zu erzeugen. Bei jedem Zertifizierungsantrag ist ein neues Schlüsselpaar zu erzeugen, dies gilt auch bei Online-Folgeanträgen.

An die Erzeugung der Teilnehmerschlüssel sind hohe Sicherheitsanforderungen zu stellen. Für die Speicherung muss die Vertraulichkeit des privaten Schlüssels gewährleistet sein. Dazu muss der private Schlüssel mindestens durch ein sicheres Passwort gesichert sein. Die Definition eines sicheren Passwortes wird aus dem Baustein ORP.4 „Identitäts- und Berechtigungsmanagement“ des BSI IT-Grundschutz-Kompendiums inklusive der Umsetzungs-Hinweise übernommen. Insbesondere „ORP.4.A22 Regelung zur Passwortqualität“ ist zu berücksichtigen.

Der private Schlüssel ist in einer PSE (Personal Security Environment) sicher zu speichern.

5.7.1 Certification Request

Eine Zertifizierungsanforderung/Zertifizierungsanfrage (ein in MIME eingebettetes PKCS#10-Objekt) kann von einem Teilnehmer, einer RA oder einer CA erstellt werden.

Die Zertifikatsanforderung (Certification Request) besteht aus einer elektronischen Anforderung in Form eines Certification Requests, den der Teilnehmer durch seine Komponenten, erstellen lassen kann und der an die CA gesendet wird. Durch die Anforderung/Anfrage wird das Zertifikat spezifiziert, welches von der CA generiert werden soll.

Für den Zertifizierungsprozess werden zwei PKI-Nachrichtentypen ausgetauscht. Die erste davon stellt die erwähnte Zertifizierungs-Anfrage dar, mit der das zu erstellende Zertifikat spezifiziert wird (self-signed). Die zweite Nachricht (Antwort auf die Anfrage/Anforderung) enthält u. a. das erstellte Zertifikat bzw. eine Fehlermeldung.

Um die zur Zertifikatsbildung notwendigen Informationen auch über die Schlüsselinformation zu sichern, sind die Anforderungen mit den folgenden Daten zu belegen:

Ein Certification Request besteht aus drei Teilen:

- CertificationRequestInfo,
- Ein Identifier für den Signatur-Algorithmus,
- Eine elektronische Signatur des Antragstellers auf der CertificationRequestInfo.

Ein CertificationRequestInfo wiederum besteht aus:

- Der Versionsnummer der verwendeten PKCS-Version,
- Dem Distinguished Name des Antragstellers,
- Dem öffentlichen Schlüssel des Antragstellers (inklusive PK-Algorithmus),
- Eine Menge von Attributen.

Attribute werden verwendet:

- Zur Angabe von zusätzlichen Informationen über den Antragsteller, die die Certification Authority (CA) benötigt, um die Zertifizierungsanfrage bearbeiten zu können (z. B. die Adresse, an die das Zertifikat zugestellt werden soll).
- Um Attribute zu spezifizieren, die in dem zu erstellenden X.509 Zertifikat erhalten sein sollen.

Eine Zertifikatserstellung und Verteilung durch die CA für den anfordernden Teilnehmer erfolgt nur unter den Bedingungen:

- Erfolgreiche Validierung des Certification Request,
- erfolgreiche Verifizierung der Signatur bzw. des Hashwertes des öffentlichen Schlüssels.

Der Ablauf ergibt sich aus der Anwendung von X.509 und PKCS#10.

5.7.2 Definition von Zertifikatsanfragen nach PKCS#10

Anhand des Standards PKSC#10 wird ein Profil erarbeitet.

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 5.8 beschrieben.

5.7.3 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 technische Profil für X.509v3-Zertifikate ist im Kapitel 4.4 beschrieben.

5.8 PKCS#10-ZERTIFIZIERUNGSANFRAGE

5.8.1 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.

5.8.2 Aufbau des PKCS#10-Datentyps

Der Datentyp eines PKCS#10-Request ist mit dem ASN.1-Datentyp CertificationRequest in [CommonPKI-2] Tabelle 1 und in [PKCS#10] in Abschnitt 4.2 festgelegt:

aus [CommonPKI-2] Tabelle 1:

1	certificationRequest-Info	DER-kodierte Anfrage-Information, bestehend aus:
1.1	version	Versionsnummer
1.2	Subject	DN des Antragstellers
1.3	subjectPublicKeyInfo	Informationen über den öffentlichen Schlüssel, der zu zertifizieren ist
1.4	attributes	Satz an Erweiterungen zum Feld “subject”
1.4.1	ExtensionReq	Erweiterung, die es erlaubt, eine oder mehr Attribute nach Standard-X.509v3-Erweiterungen hinzuzufügen
2	signatureAlgorithm	Kennzeichen für den Signaturalgorithmus, mit dem CertificationRequestInfo signiert wird.
2.1	Signature	Ergebnis des Signierens von CertificationRequestInfo, 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 4.4.11 „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 RSAES-PSS einzusetzen ist; vgl. Abschnitt 2.1.2. Die Datenfelder sind entsprechend dem verwendeten Algorithmus zu belegen (vgl. Abschnitt 2.1.2.1).

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

5.8.3 Aufbau der Teildatenstruktur CertificationRequestInfo

5.8.3.1 Versionsnummer

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

> **Profilierung:**
> In [CommonPKI-2], 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".

5.8.3.2 Namen von Zertifikatsinhabern

Das Datenfeld subject in einem PKCS#10-Datentyp entspricht dem gleichnamigen Datenfeld in einem X.509v3-Zertifikat; vgl Abschnitt 4.4.5.

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 4.4.5 zum DN für Teilnehmerzertifikate sind verbindlich.

5.8.3.3 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.

5.8.3.4 Profilierung

Die Angaben in Abschnitt Abschnitt 4.4.7 „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:

BasicConstrints::= SEQUENCE {
CA {TRUE, FALSE}
pathLenConstrint OCTET STRING}
Öffentlicher Schlüssel des Zertifikatsinhabers

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

5.8.3.5 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, sind die Datenfelder attributes, insbesondere ExtensionReq nicht zu füllen.

Profilierung: Die Zertifizierungsanfrage soll keine Erweiterungen enthalten.

5.8.4 Transport der PKCS#10 Zertifizierungsanfrage

5.8.4.1 Transportformat

Referenz: [CommonPKI-2], Abschnitt 2.3
Abschnitte 1.4

Nachdem ein Datenobjekt vom Typ certificationRequestInfo (vgl. 5.8.2 „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 re-

sultierende 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.

5.8.4.2 Transportwege

Es muss beachtet werden, dass jedes Trust Center nur bestimmte Transportwege anbietet, unabhängig der möglichen Transportwege in den Anlagen der Gemeinsamen Grundsätze Technik. Die Spezifikationen zu den Transportwegen sind auf den Webseiten der Trust Center (z. B. www.trust-center.info, www.dktig.de) veröffentlicht.

5.9 PKCS#7-ZERTIFIZIERUNGSANTWORT

Referenz: [CommonPKI-2], Abschnitt 2.3

5.9.1 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#710 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.

[10] 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.

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
SignedAndEnvelopedData	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

5.9.2 Aufbau der PKCS#7-Zertifizierungsantwort

Eine PKCS#7-Zertifizierungsantwort nach [CommonPKI-2] ist ein CMS-Datenobjekt vom „Inhalts"-Typ SignedData.

Profilierung:
[CommonPKI-2] 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 [CommonPKI-2], 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 [CommonPKI-2], 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 [CommonPKI-2] 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.

5.9.3 Aufbau der Teildatenstruktur Content vom ContentType „SignedData“

5.9.3.1 Datenfeld “version”

Das Feld version enthält die Versionsnummer der Syntax. Die Versionsnummer ist je nach Art der zu signierenden Daten entweder „1“ oder „3“11.

> **Profilierung:**
> Für das Datenfeld version soll in dieser Datenstruktur immer der Wert „1“ eingesetzt werden.

5.9.3.2 Datenfeld “digestAlgorithms”

Dieses Datenfeld enthält genau einen Objektbezeichner12 für den Hash-Algorithmus, der zur Signatur von content eingesetzt wird.

> Der zu verwendende Objektbezeichner ist von [CommonPKI-2] Tabelle 2 auf OID 2.16.840.1.101.3.4.2.1 für SHA-256 festgelegt; vgl. Abschnitt 2.1.1

5.9.3.3 Datenfeld „encapContentInfo“

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

> - **Profilierung:**
>   Gemäß [CommonPKI-2], Abschnitt 2.1.1 entfallen die Datenfelder encapContent und signerInfos für eine Zertifizierungsantwort aus dem CMS-Datenobjekt.

[11] 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.

[12] 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.

5.9.3.4 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 [CommonPKI-2] 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.

5.9.3.5 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. Die Sperrlisten werden in einem Verzeichnis zur Verfügung gestellt.

5.9.3.6 Datenfeld “signerInfos”

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

> **Profilierung:**
> Gemäß [CommonPKI-2], Abschnitt 2.1.1 entfallen die Datenfelder encapContent und signerInfos für eine Zertifizierungsantwort aus dem CMS-Datenobjekt.

5.9.4 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

5.9.4.1 Transportformat..

Referenz: [CommonPKI-2], Abschnitt 2.3
Abschnitte 1.4

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.

5.9.4.2 Transportwege

Für den Transport von PKCS#7-Zertifizierungsantwort zu den Antragstellern gelten die gleichen Transportwege wie bei der Antragsstellung.

5.9.5 Sperrlisten

Die Sperrlisten werden in regelmäßigen Abständen (bei jeder Änderung) innerhalb des Verzeichnisdienstes veröffentlicht. Unter Bezug auf das Gültigkeitsmodell ist der aktuelle Abruf der Sperrlisten notwendig (Empfehlung – tagesaktueller Abruf).

Die Verarbeitung der jeweils aktuellen Sperrlisten wird vorausgesetzt. Die Definition der Profile für Zertifikate und Sperrlisten entspricht den MTTv2-Spezifikationen.

Analog zu dem vorgesehenen Gültigkeitsmodell ist die Gültigkeit von Schlüssel und Zertifikaten entsprechend MTTv2 als sogenanntes „Schalenmodell“ ausreichend definiert.

Auf dieser Grundlage haben die Teilnehmer auf eigenes Risiko zu bestimmen, ob und in welchen Intervallen Sperrlisten herangezogen werden.

Verarbeitung von Sperrlisten
Jeder Teilnehmer bzw. UA muss die Verarbeitung von Sperrlisten (Certificate Revocation List, CRL) unterstützen. Dazu sind die folgenden Funktionalitäten bereitzustellen:

- Anforderung zur Sperrung eines Zertifikats,
- Anforderung von Sperrlisten von einer CA,

- Echtheits-Verifizierung von Sperrlisten (Sperrlisten sind von der CA signiert),
- Abgleich der Sperrliste mit lokaler Zertifikatsliste (Adressliste),
- periodisches Überprüfen des Gültigkeitszeitraums einer Sperrliste (dadurch kann das Anfordern einer neuen Sperrliste ausgelöst werden).

6. Anhang

Um Kompatibilität zwischen den verschiedenen Teilnehmern erreichen zu können, müssen neben den kryptographischen Sicherheitsverfahren auch die Strukturen sicherheitsrelevanter Daten so weit wie nötig festgelegt werden. Diese Strukturen sind kompatibel zu den Datenstrukturen und Zertifikaten nach INTERNET-Konventionen (diverse RFC) sowie nach ITU-T (X.500-Serie) festzulegen.

6.1 ASN.1 Syntax relevante Datenstrukturen

6.1.1 Öffentlicher und privater Schlüssel nach X.509

Öffentliche und private RSA-Schlüssel haben die folgende Syntax:

```
RSAPublicKey :: =       SEQUENCE {
    modulus             INTEGER,        /*  n   */
    publicExponent      INTEGER }       /*  e   */

RSAPrivateKey :: =      SEQUENCE {
    modulus             INTEGER,        /*  n   */
    secretExponent      INTEGER }       /*  d   */
```

Alternativ dazu, kann das Format des privaten Schlüssels auch wie folgt realisiert werden:

```
RSAPrivateKey :: =      SEQUENCE {
    prime1              INTEGER,        /*  p   */
    prime2              INTEGER }       /*  q   */
```

In diesem Fall kann mit den Strukturen für den öffentlichen und privaten Schlüssel bei Bedarf eine an PKCS angelehnte Struktur erzeugt werden. Der erforderliche Speicherplatz lässt sich gegenüber diesem Format minimieren. Dieser Effekt ist besonders dann von Vorteil, wenn die Implementierung auf Systemen erfolgen muss, die relativ wenig Speicherplatz zur Verfügung haben (z. B. Chipkarten). Bei Anwendung des privaten Schlüssels sind allerdings die zugehörigen Parameter - insbesondere der geheime Exponent - erst zu erzeugen.

6.1.2 X.509v3-Zertifikat, Zertifizierungspfad

X.509-Zertifikate sowie Zertifizierungspfade werden durch die folgenden Strukturen in ASN.1 Syntax definiert:

```
Certificate :: =                SIGNED SEQUENCE {
    version [0]                 Version DEFAULT v1988,
    serialNumber                CertificateSerialNumber,
    signature                   AlgorithmIdentifier,
    issuer                      Name,
    validity                    Validity,
    subject                     Name,
    subjectPublicKeyInfo        SubjectPublicKeyInfo }

IssuerUniqueIdentifier          Name, {OPTIONAL}
SubjectUniqueIdentifier         Name, {OPTIONAL}

Version ::=                     INTEGER { v1988(0) }  (Version 1 oder 3)

CertificateSerialNumber ::=     INTEGER
Validity ::=                    SEQUENCE {
    notBefore                   GeneralizedTime,
    notAfter                    GeneralizedTime }
SubjectPublicKeyInfo ::=        SEQUENCE {
    algorithm                   AlgorithmIdentifier,
    subjectPublicKey            BIT STRING }

Certification Extensions
AlgorithmIdentifier ::=         SEQUENCE {
    algorithm                   OBJECT IDENTIFIER,
    parameters                  ANY DEFINED BY algorithm OPTIONAL }
Certificates ::=                SEQUENCE {
    certificate                 Certificate,
    certificationPath           ForwardCertificationPath OPTIONAL }
```

6.1.3 Sperrliste

Die folgende ASN.1 Syntax definiert das Format einer Sperrliste:

```
CertificateRevocationList :: =  SIGNED SEQUENCE {
    signature                   AlgorithmIdentifier,
    issuer                      Name,
```

```
        lastUpdate              GeneralizedTime,
        nextUpdate              GeneralizedTime,
        revokedCertificates     SEQUENCE OF CRLEntry OPTIONAL }

    CRLEntry ::=                SEQUENCE {
        userCertificate         SerialNumber,
        revocationDate          GeneralizedTime }
```

6.2 ASN.1 Syntax relevanter Makros

6.2.1 Signierte Struktur

Die folgende ASN.1 Syntax eines Makros definiert das Format einer signierten Struktur:

```
    SIGNED MACRO  ::=
    BEGIN

    TYPE NOTATION ::=           type (ToBeSigned)

    VALUE NOTATION    ::=       value    (VALUE

        SEQUENCE {
            ToBeSigned,
            AlgorithmIdentifier (des Signaturverfahrens),
            ENCRYPTED OCTET STRING (OCTET STRING ist der Hashwert
            vom Datenelement "ToBeSigned") }
        )
    END         of SIGNED
```

6.2.2 ASN.1 Syntax einer Signatur

Die folgende ASN.1 Syntax definiert das Format einer Signatur ;

```
    SIGNATURE MACRO ::=
    BEGIN

    TYPE NOTATION ::=           type (ofSignature)

    VALUE NOTATION    ::=       value    (VALUE

        SEQUENCE {
            AlgorithmIdentifier (des Signaturverfahrens),
```

```
            ENCRYPTED OCTET STRING (OCTET STRING ist der Hashwert
            vom Datenelement "ofSignature") }
        )
    END     of SIGNATURE
```

6.3 Kommunikationssystem

6.3.1 Grundsatz

Die für das Routing der Daten erforderlichen Informationen sind zu liefern. Im Rahmen des Datenaustausches werden zwischen zwei Kommunikationspartnern Nutzdatendateien ausgetauscht. Dabei können, in Abhängigkeit der vorhandenen Übertragungswege eine oder mehrere Stellen als Vermittlungsstellen fungieren. Unabhängig von der Art der Daten sollen die kommunizierenden Stellen die notwendigen Informationen erhalten, die es erlauben, Nutzdaten ohne Kenntnis der eigentlichen Dateninhalte zu befördern.

6.3.2 Voraussetzungen und Forderungen für den Datenaustausch signierter und verschlüsselter Datenobjekte (Datenträger und sonstige Datenfernübertragungsverfahren)

Zur Nutzung von Übertragungsprotokollen (HTTP, FTP u.s.w.) sowie alternativen Datenfernübertragungsverfahren z. B. auf Datenträgern werden Datenobjekte entsprechend der PKCS#7-Syntax signiert und verschlüsselt.

Die Organisationen der Beteiligten im Datenaustausch sieht vor, dass Datenpakete auch über Dritte (sog. Weiterleitungsstellen) vermittelt werden, die nicht befugt sind, die Nutzdaten zu entschlüsseln. Dementsprechend müssen die Transportinformationen begleitend zu den Nutzdaten unverschlüsselt übermittelt werden.

Die signierten und verschlüsselten Nutzdaten werden von einer Auftragssatzdatei (siehe Anlage 2 der Gemeinsamen Grundsätzen Technik) begleitet, die alle relevanten Transportinformationen in unverschlüsselter Form enthält.

Die Rahmenbedingungen werden in der jeweils geltenden Fassung der Gemeinsamen Grundsätzen Technik beschrieben

6.4 Beispiele

6.4.1 Struktur einer Schlüsselliste gemäß Kapitel 4.6.1

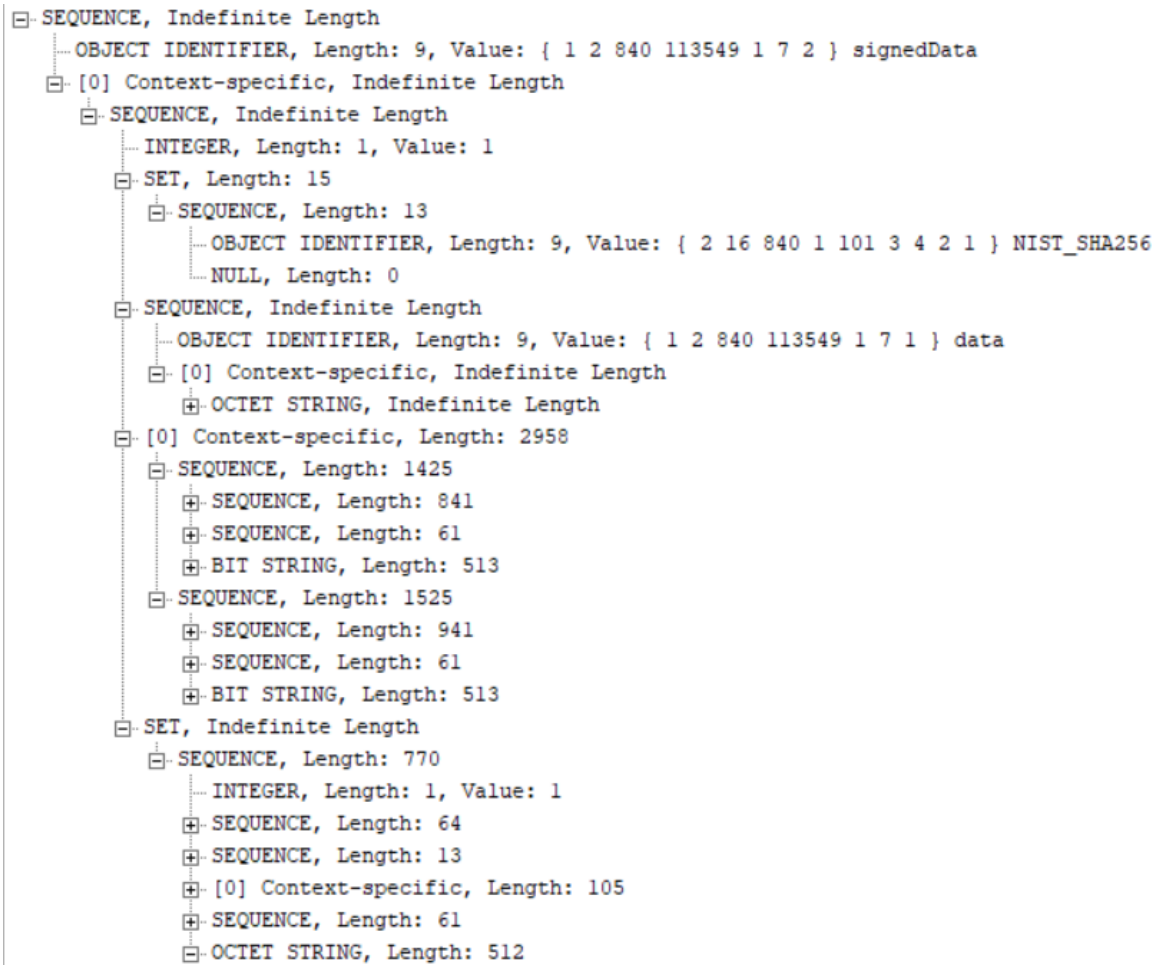
Die Struktur einer Gesamtliste mit allen öffentlichen Schlüsseln ist anhand der nachfolgenden Auflistung beispielhaft dargestellt:

- PCA-Schlüssel mit Serien-Nummer 0001
- CA-Schlüssel mit Serien-Nummer 0010 (z.B. ITSG Trust Center)
- Teilnehmerschlüssel ausgestellt von der CA mit Serien-Nummer 0001
- Teilnehmerschlüssel ausgestellt von der CA mit Serien-Nummer 0002
- Teilnehmerschlüssel ausgestellt von der CA mit Serien-Nummer 0003
- …. Serien-Nummer 000x
- …. Serien-Nummer 000y
- …. Serien-Nummer 000z
- CA-Schlüssel mit Serien-Nummer 0011 (z.B. DKTIG Trust Center)
- Teilnehmerschlüssel ausgestellt von der CA mit Serien-Nummer 0001
- Teilnehmerschlüssel ausgestellt von der CA mit Serien-Nummer 0002
- Teilnehmerschlüssel ausgestellt von der CA mit Serien-Nummer 0003
- …. Serien-Nummer 000x
- …. Serien-Nummer 000y
- …. Serien-Nummer 000z
- PCA-Schlüssel mit Serien-Nummer 0002
- CA-Schlüssel mit Serien-Nummer 0012 (z.B. ITSG Trust Center)
- Teilnehmerschlüssel ausgestellt von der CA mit Serien-Nummer 0001
- Teilnehmerschlüssel ausgestellt von der CA mit Serien-Nummer 0002
- Teilnehmerschlüssel ausgestellt von der CA mit Serien-Nummer 0003
- …. Serien-Nummer 000x
- …. Serien-Nummer 000y
- …. Serien-Nummer 000z
- CA-Schlüssel mit Serien-Nummer 0013 (z.B. DKTIG Trust Center)
- Teilnehmerschlüssel ausgestellt von der CA mit Serien-Nummer 0001
- Teilnehmerschlüssel ausgestellt von der CA mit Serien-Nummer 0002
- Teilnehmerschlüssel ausgestellt von der CA mit Serien-Nummer 0003
- …. Serien-Nummer 000x
- …. Serien-Nummer 000y
- …. Serien-Nummer 000z

6.4.2 Beispiel einer signierten Nachricht gemäß Kapitel 3.2

Das hier gezeigte Beispiel einer PKCS#7-signierten Nachricht wird mit dem ASN.1 Anzeigetool BERViewer dargestellt und kommentiert.

Übersicht einer signierten Nachricht



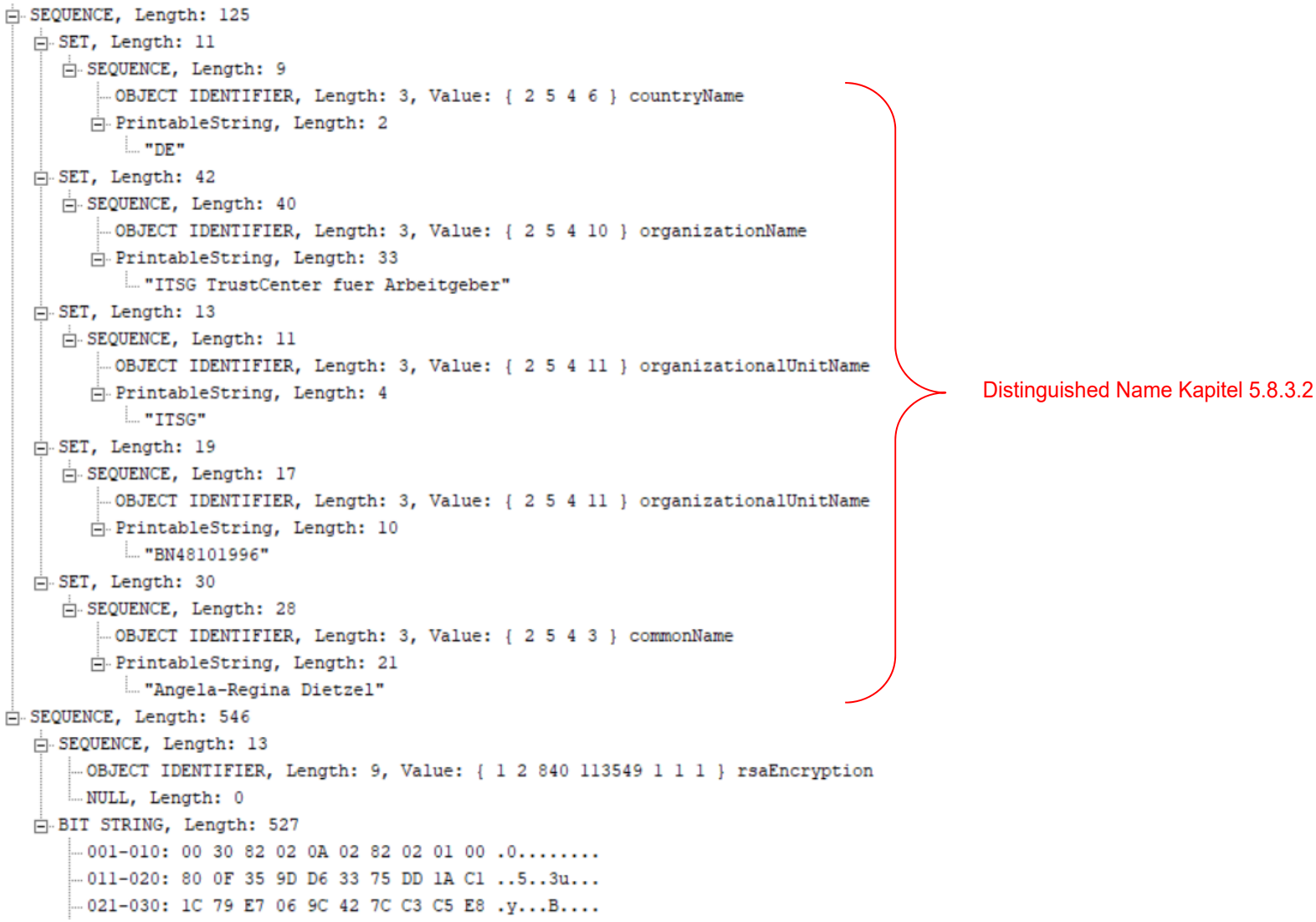
Teil 1

```
SEQUENCE, Indefinite Length
    OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 7 2 } signedData
    [0] Context-specific, Indefinite Length
        SEQUENCE, Indefinite Length
            INTEGER, Length: 1, Value: 1
            SET, Length: 15
                SEQUENCE, Length: 13
                    OBJECT IDENTIFIER, Length: 9, Value: { 2 16 840 1 101 3 4 2 1 } NIST_SHA256
                    NULL, Length: 0
            SEQUENCE, Indefinite Length
                OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 7 1 } data
                [0] Context-specific, Indefinite Length
                    OCTET STRING, Indefinite Length
                        OCTET STRING, Length: 60
                            01-10: 53 69 67 6E 69 65 72 20 54 65 Signier Te
                            11-20: 73 74 2E 0D 0A 0D 0A 44 69 65 st.....Die
                            21-30: 73 65 20 54 65 78 74 20 44 61 se Text Da
                            31-40: 74 65 69 20 68 69 65 72 20 73 tei hier s
                            41-50: 6F 6C 6C 20 73 69 67 6E 69 65 oll signie
                            51-60: 72 74 20 77 65 72 64 65 6E 2E rt werden.
```

Teil 2

```
[0] Context-specific, Length: 2958
    SEQUENCE, Length: 1425
        SEQUENCE, Length: 841
            [0] Context-specific, Length: 3
                INTEGER, Length: 1, Value: 2
            INTEGER, Length: 3, Value: 1000049
            SEQUENCE, Length: 61
                OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 1 10 } rsaPSS
                SEQUENCE, Length: 48
                    [0] Context-specific, Length: 13
                        SEQUENCE, Length: 11
                            OBJECT IDENTIFIER, Length: 9, Value: { 2 16 840 1 101 3 4 2 1 } NIST_SHA256
                    [1] Context-specific, Length: 26
                        SEQUENCE, Length: 24
                            OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 1 8 } rsaMGF1
                            SEQUENCE, Length: 11
                                OBJECT IDENTIFIER, Length: 9, Value: { 2 16 840 1 101 3 4 2 1 } NIST_SHA256
                    [2] Context-specific, Length: 3
                        INTEGER, Length: 1, Value: 32
            SEQUENCE, Length: 57
                SET, Length: 11
                    SEQUENCE, Length: 9
                        OBJECT IDENTIFIER, Length: 3, Value: { 2 5 4 6 } countryName
                        PrintableString, Length: 2
                            "DE"
                SET, Length: 42
                    SEQUENCE, Length: 40
                        OBJECT IDENTIFIER, Length: 3, Value: { 2 5 4 10 } organizationName
                        PrintableString, Length: 33
                            "ITSG TrustCenter fuer Arbeitgeber"
            SEQUENCE, Length: 30
                UTCTime, Length: 13, Value: "181114000000Z"
                UTCTime, Length: 13, Value: "211231235959Z"
```

Teil 3



Distinguished Name Kapitel 5.8.3.2

Teil 4

```
SEQUENCE, Length: 61
    OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 1 10 } rsaPSS
    SEQUENCE, Length: 48
        [0] Context-specific, Length: 13
            SEQUENCE, Length: 11
                OBJECT IDENTIFIER, Length: 9, Value: { 2 16 840 1 101 3 4 2 1 } NIST_SHA256
        [1] Context-specific, Length: 26
            SEQUENCE, Length: 24
                OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 1 8 } rsaMGF1
                SEQUENCE, Length: 11
                    OBJECT IDENTIFIER, Length: 9, Value: { 2 16 840 1 101 3 4 2 1 } NIST_SHA256
        [2] Context-specific, Length: 3
            INTEGER, Length: 1, Value: 32
BIT STRING, Length: 513
    001-010: 00 A0 35 BF 00 90 FA D9 2F 31 ..5...../1
    011-020: C2 16 F2 D5 C9 73 4A 7E 98 E4 .....sJ...
    021-030: 69 4D 9D 3C 2E 12 53 01 1F 3A iM.<..S..:
    031-040: 25 A9 D7 A5 50 B8 F3 D6 37 D7 %...P...7.
```

Teil 5

```
SEQUENCE, Length: 1525
    SEQUENCE, Length: 941
        [0] Context-specific, Length: 3
            INTEGER, Length: 1, Value: 2
        INTEGER, Length: 1, Value: 71
        SEQUENCE, Length: 61
            OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 1 10 } rsaPSS
            SEQUENCE, Length: 48
                [0] Context-specific, Length: 13
                    SEQUENCE, Length: 11
                        OBJECT IDENTIFIER, Length: 9, Value: { 2 16 840 1 101 3 4 2 1 } NIST_SHA256
                [1] Context-specific, Length: 26
                    SEQUENCE, Length: 24
                        OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 1 8 } rsaMGF1
                        SEQUENCE, Length: 11
                            OBJECT IDENTIFIER, Length: 9, Value: { 2 16 840 1 101 3 4 2 1 } NIST_SHA256
                [2] Context-specific, Length: 3
                    INTEGER, Length: 1, Value: 32
        SEQUENCE, Length: 70
            SET, Length: 11
                SEQUENCE, Length: 9
                    OBJECT IDENTIFIER, Length: 3, Value: { 2 5 4 6 } countryName
                    PrintableString, Length: 2
                        "DE"
            SET, Length: 55
                SEQUENCE, Length: 53
                    OBJECT IDENTIFIER, Length: 3, Value: { 2 5 4 10 } organizationName
                    PrintableString, Length: 46
                        "Datenaustausch im Gesundheits- und Sozialwesen"
        SEQUENCE, Length: 30
            UTCTime, Length: 13, Value: "1712181258352"
            UTCTime, Length: 13, Value: "2302152359592"
```

Teil 6

```
SEQUENCE, Length: 57
  SET, Length: 11
    SEQUENCE, Length: 9
      OBJECT IDENTIFIER, Length: 3, Value: { 2 5 4 6 } countryName
      PrintableString, Length: 2
        "DE"
  SET, Length: 42
    SEQUENCE, Length: 40
      OBJECT IDENTIFIER, Length: 3, Value: { 2 5 4 10 } organizationName
      PrintableString, Length: 33
        "ITSG TrustCenter fuer Arbeitgeber"
SEQUENCE, Length: 546
  SEQUENCE, Length: 13
    OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 1 1 } rsaEncryption
    NULL, Length: 0
  BIT STRING, Length: 527
    001-010: 00 30 82 02 0A 02 82 02 01 00 .0........
    011-020: FB 91 07 93 60 CB 96 0D 33 F7 ....`...3.
    021-030: D0 9F 28 DE C6 44 8B 24 35 C5 ..(..D.$5.
    031-040: D7 20 70 69 C7 4C AE 59 F5 93 . pi.L.Y..
    041-050: 2B D8 8E 5B 02 33 6E 78 EF BC +..[.3nx..
```

Teil 7

```
[3] Context-specific, Length: 154
  SEQUENCE, Length: 151
    SEQUENCE, Length: 15
      OBJECT IDENTIFIER, Length: 3, Value: { 2 5 29 19 } basicConstraints
      OCTET STRING, Length: 8
        1-8: 30 06 01 01 FF 02 01 00       0.......
    SEQUENCE, Length: 11
      OBJECT IDENTIFIER, Length: 3, Value: { 2 5 29 15 } keyUsage
      OCTET STRING, Length: 4
        1-4: 03 02 01 06                   ....
    SEQUENCE, Length: 29
      OBJECT IDENTIFIER, Length: 3, Value: { 2 5 29 14 } subjectKeyIdentifier
      OCTET STRING, Length: 22
        01-10: 04 14 43 69 9D EF 73 78 BD 4E ..Ci..sx.N
        11-20: AE F9 94 21 43 65 F2 1A DE FE ...!Ce....
        21-22: BA D3                         ..
    SEQUENCE, Length: 88
      OBJECT IDENTIFIER, Length: 3, Value: { 2 5 29 35 } authorityKeyIdentifier
      OCTET STRING, Length: 81
        01-10: 30 4F A1 4A A4 48 30 46 31 0B 0O.J.H0F1.
        11-20: 30 09 06 03 55 04 06 13 02 44 0...U....D
        21-30: 45 31 37 30 35 06 03 55 04 0A E1705..U..
```

Teil 8

```
SEQUENCE, Length: 61
  OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 1 10 } rsaPSS
  SEQUENCE, Length: 48
    [0] Context-specific, Length: 13
      SEQUENCE, Length: 11
        OBJECT IDENTIFIER, Length: 9, Value: { 2 16 840 1 101 3 4 2 1 } NIST_SHA256
    [1] Context-specific, Length: 26
      SEQUENCE, Length: 24
        OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 1 8 } rsaMGF1
        SEQUENCE, Length: 11
          OBJECT IDENTIFIER, Length: 9, Value: { 2 16 840 1 101 3 4 2 1 } NIST_SHA256
    [2] Context-specific, Length: 3
      INTEGER, Length: 1, Value: 32
BIT STRING, Length: 513
  001-010: 00 57 AF 89 4B E5 1F D9 21 17 .W..K...!.
  011-020: 51 34 FE D7 BF DA A9 D4 5B EF Q4......[.
  021-030: E0 C0 32 7F F3 51 E0 60 92 8D ..2..Q.`..
  031-040: 74 FA 21 C1 6E 3B 1A D0 8F 57 t.!.n;...W
  041-050: E5 E1 E4 71 EC AB B1 A1 C6 8E ...q......
```

Teil 9

```
SET, Indefinite Length
  SEQUENCE, Length: 770
    INTEGER, Length: 1, Value: 1
    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
              "DE"
        SET, Length: 42
          SEQUENCE, Length: 40
            OBJECT IDENTIFIER, Length: 3, Value: { 2 5 4 10 } organizationName
            PrintableString, Length: 33
              "ITSG TrustCenter fuer Arbeitgeber"
        INTEGER, Length: 3, Value: 1000049
      SEQUENCE, Length: 13
        OBJECT IDENTIFIER, Length: 9, Value: { 2 16 840 1 101 3 4 2 1 } NIST_SHA256
        NULL, Length: 0
      [0] Context-specific, Length: 105
        SEQUENCE, Length: 24
          OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 9 3 } contentType
          SET, Length: 11
            OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 7 1 } data
        SEQUENCE, Length: 28
          OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 9 5 } signingTime
          SET, Length: 15
            UTCTime, Length: 13, Value: "200513101310Z"
        SEQUENCE, Length: 47
          OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 9 4 } messageDigest
          SET, Length: 34
            OCTET STRING, Length: 32
              01-10: B3 00 95 4E B5 52 3C 58 61 18 ...N.R<Xa.
              11-20: AF 50 14 B9 88 91 E5 F5 37 40 .P......7@
              21-30: 36 0B 81 17 B2 5E 11 0A 73 45 6....^..sE
              31-32: 19 96                         ..
```

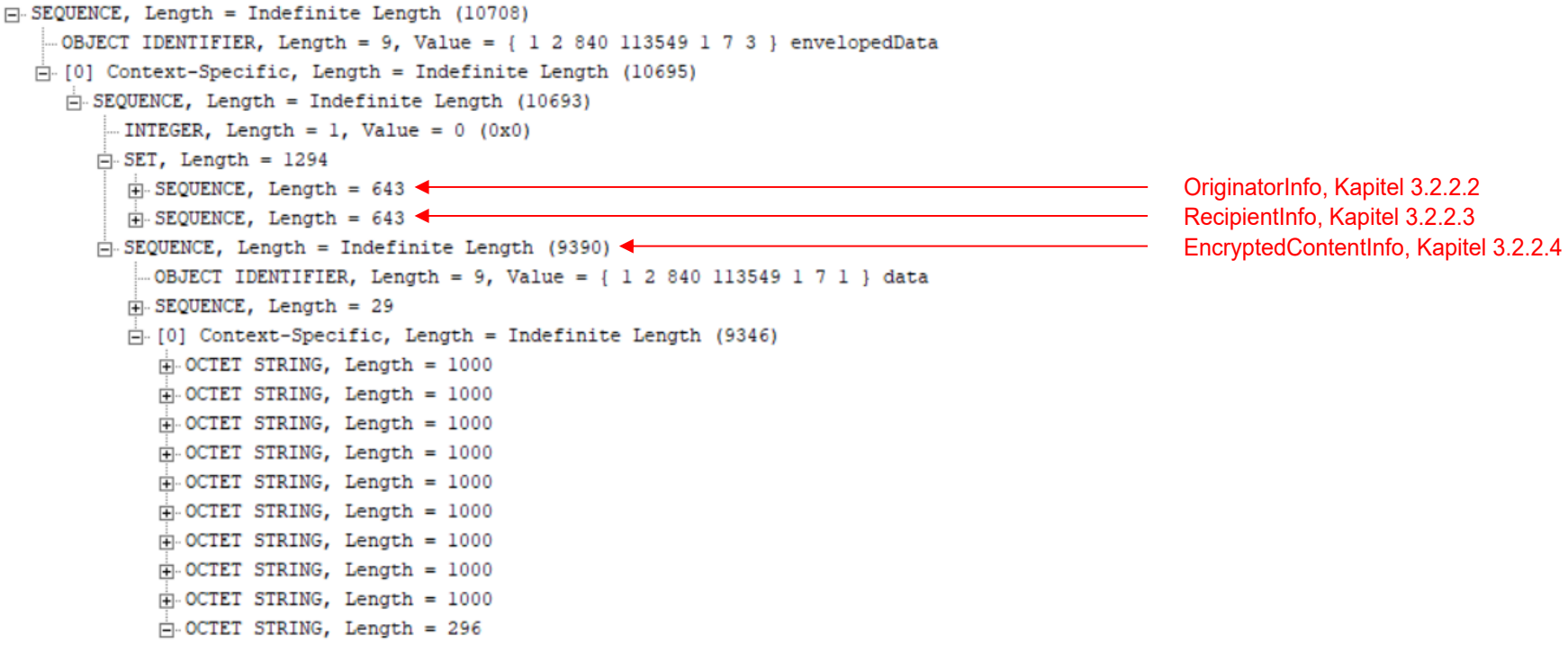
Teil 10

```
SEQUENCE, Length: 61
    OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 1 10 } rsaPSS
    SEQUENCE, Length: 48
        [0] Context-specific, Length: 13
            SEQUENCE, Length: 11
                OBJECT IDENTIFIER, Length: 9, Value: { 2 16 840 1 101 3 4 2 1 } NIST_SHA256
        [1] Context-specific, Length: 26
            SEQUENCE, Length: 24
                OBJECT IDENTIFIER, Length: 9, Value: { 1 2 840 113549 1 1 8 } rsaMGF1
                SEQUENCE, Length: 11
                    OBJECT IDENTIFIER, Length: 9, Value: { 2 16 840 1 101 3 4 2 1 } NIST_SHA256
        [2] Context-specific, Length: 3
            INTEGER, Length: 1, Value: 32
OCTET STRING, Length: 512
    001-010: 02 DC 53 D8 A2 A8 4B A9 9D 71 ..S...K..q
    011-020: A6 8C 22 E3 43 6F 4A C1 52 85 ..".CoJ.R.
    021-030: 98 12 20 15 41 5B 94 BB 2B 28 .. .A[..+(
    031-040: EB 8E 97 48 0B 2B F6 98 A3 3C ...H.+...<
    041-050: 12 62 31 EC 0C 4F D0 18 1E 7A .b1..O...z
    051-060: C5 2F 61 FE 36 0C 91 2A C1 CD ./a.6..*..
```

6.4.3 Beispiel einer PKCS#7-verschlüsselten Nachricht gemäß Kapitel 3.2.2

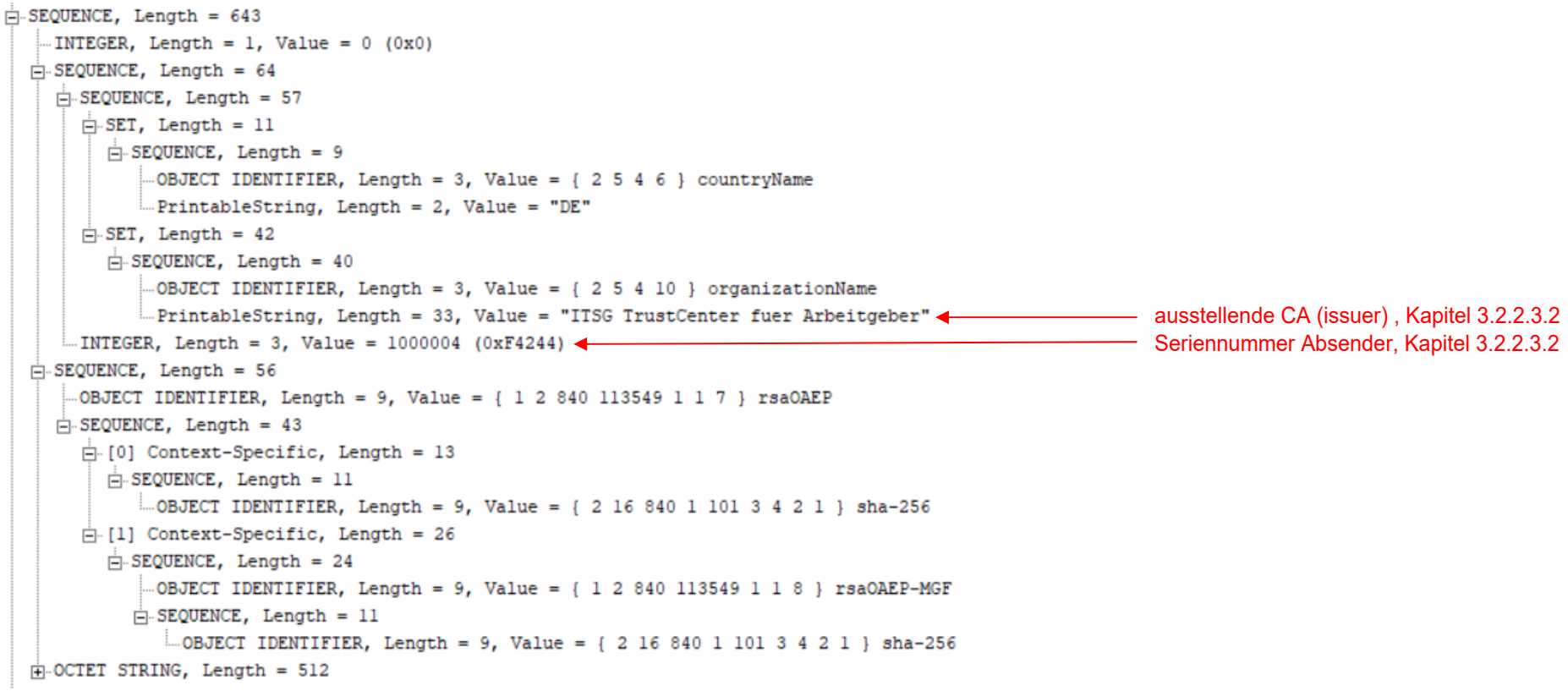
Das hier gezeigte Beispiel einer PKCS#7-verschlüsselten Nachricht mit RSA keyEncryption und AES ContentEncryption ist auf die drei wesentlichen Sequenzen zusammengezogen und entsprechend kommentiert.

Übersicht einer verschlüsselten Nachricht



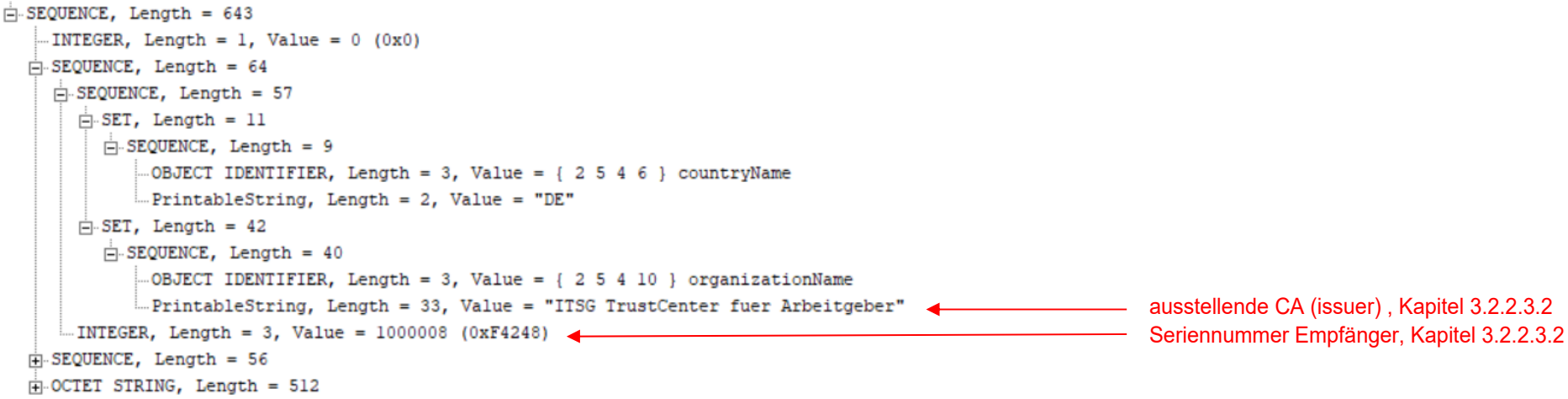
Teil 1, OriginatorInfo

Hier werden die wesentlichen Details in den beiden Sequenzen für Absender (Originator) gelistet.

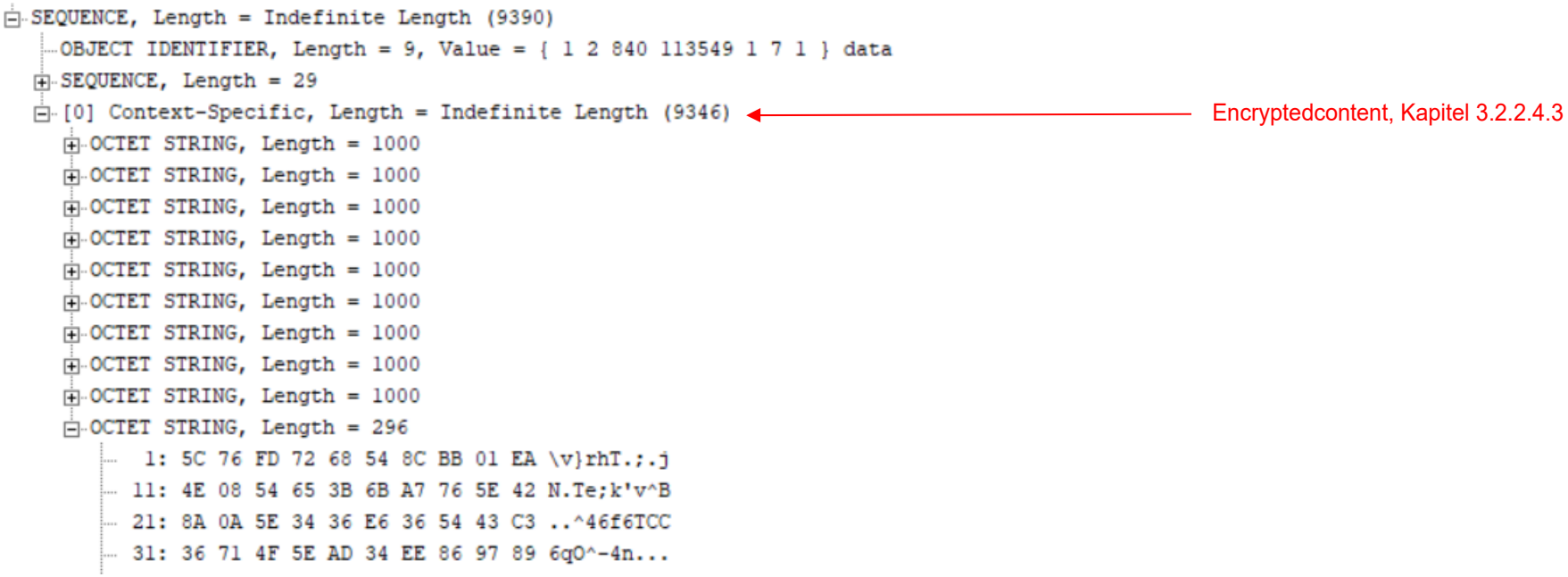


Teil 2 ReciepientInfo

Hier werden die wesentlichen Details in den beiden Sequenzen für und Empfänger (Recipient) gelistet.



Teil 3, EncryptedContentInfo



6.4.4 Beispiel einer PKCS#10 Zertifizierungsanfrage gemäß Kapitel 5.8

Das hier gezeigte Beispiel einer PKCS#10 Zertifizierungsanfrage wird mit dem ASN.1 Anzeigetool BERViewer dargestellt und entsprechend kommentiert.
Die Zertifizierungsanfrage ist zur besseren Übersichtlichkeit in zwei Teile aufgegliedert.

Teil 1

```
SEQUENCE, Length = 1266
  SEQUENCE, Length = 682
    INTEGER, Length = 1, Value = 0 (0x0)
    SEQUENCE, Length = 125
      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"
      SET, Length = 18
        SEQUENCE, Length = 16
          OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 11 } organizationalUnitName
          PrintableString, Length = 9, Value = "ITSG GmbH"
      SET, Length = 19
        SEQUENCE, Length = 17
          OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 11 } organizationalUnitName
          PrintableString, Length = 10, Value = "BN99301176"
      SET, Length = 25
        SEQUENCE, Length = 23
          OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 3 } commonName
          PrintableString, Length = 16, Value = "Herr Udo Mueller"
    SEQUENCE, Length = 546
      SEQUENCE, Length = 13
        OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 1 } rsaEncryption
        NULL
      BIT STRING, Length = 527, Unused bits = 0
          1: 30 82 02 0A 02 82 02 01 00 80 0.........
         11: 3A 92 4A 23 C9 00 7F 57 3F 2A :.J#I..W?*
         21: 9F 13 EC 72 1B D7 01 3C 4B E2 ..1r.W.<Kb
         31: C7 1C 82 98 69 E4 AC 0E DD 95 G...id,.].
```

Distinguished Name Kapitel 5.8.3.2

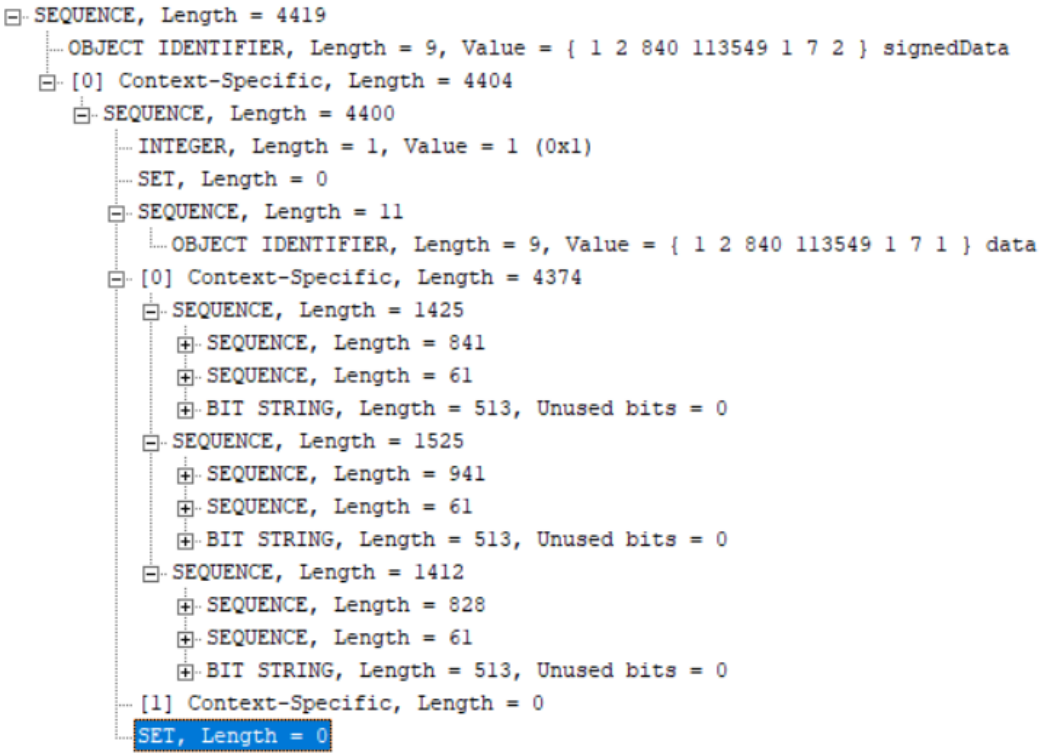
Teil 2

```
    [0] Context-Specific, Length = 0
SEQUENCE, Length = 61
    OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 10 } rsaPSS
  SEQUENCE, Length = 48
    [0] Context-Specific, Length = 13
      SEQUENCE, Length = 11
          OBJECT IDENTIFIER, Length = 9, Value = { 2 16 840 1 101 3 4 2 1 } sha-256
    [1] Context-Specific, Length = 26
      SEQUENCE, Length = 24
          OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 8 } rsaOAEP-MGF
        SEQUENCE, Length = 11
            OBJECT IDENTIFIER, Length = 9, Value = { 2 16 840 1 101 3 4 2 1 } sha-256
    [2] Context-Specific, Length = 3
        INTEGER, Length = 1, Value = 32 (0x20)
BIT STRING, Length = 513, Unused bits = 0
    1: 2A 5A 00 20 47 14 E0 9E 42 12 *Z. G.`.B.
   11: 94 02 B7 C0 9F 3F 62 6B C3 BD ..7@.?bkC=
   21: 89 E5 9F 91 97 84 AB 95 4A 5E .e....+.J^
   31: EB AF 56 2E 16 11 2F 9C 57 B4 k/V.../.W4
   41: 38 AA 47 F4 7C 47 98 B8 7F 88 8*Gt|G.8..
```

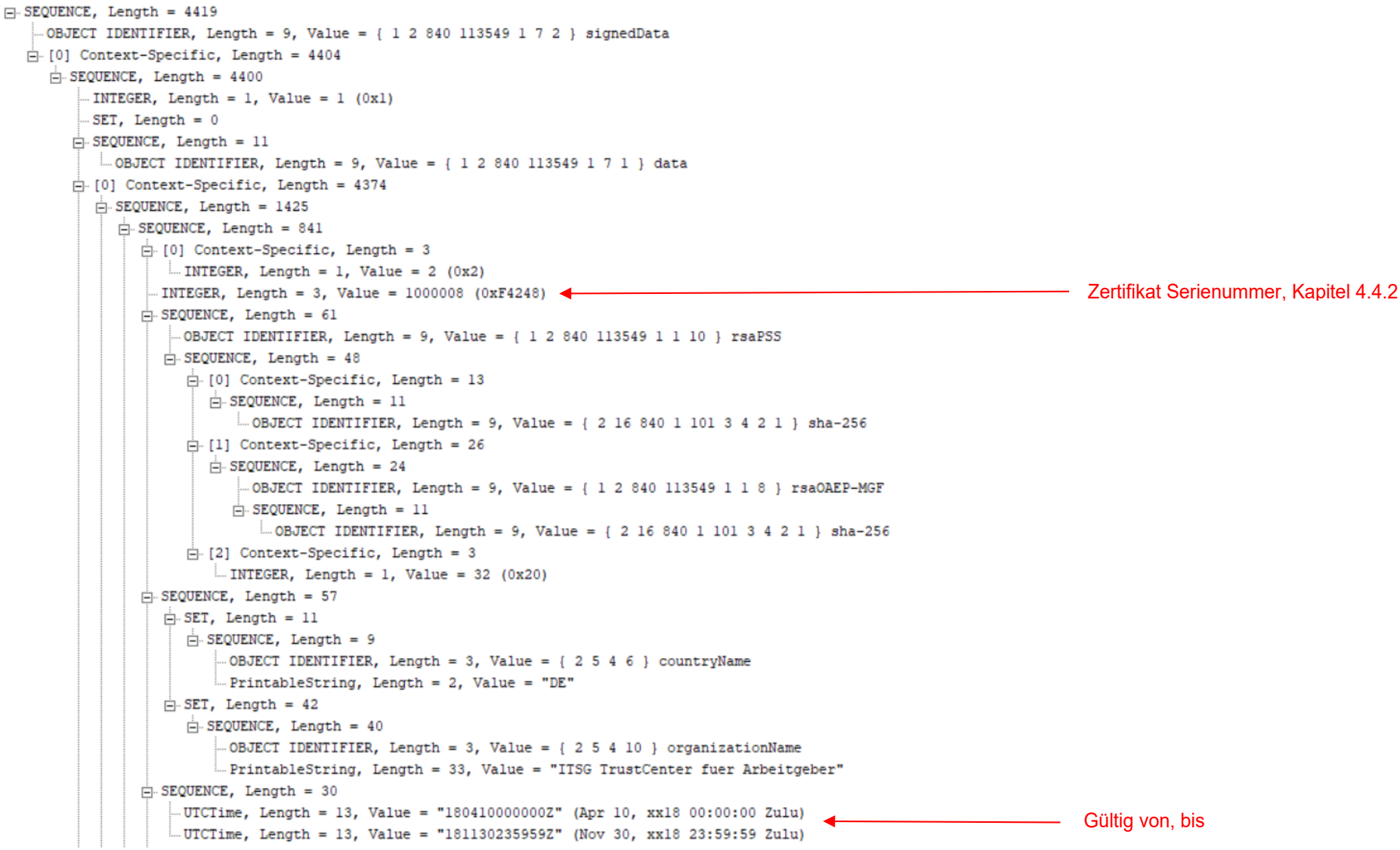
6.4.5 Beispiel einer PKCS#7 Zertifizierungsantwort gemäß Kapitel 5.9

Das hier gezeigte Beispiel einer PKCS#7 Zertifizierungsantwort mit der Hash-Variante SHA-256 wird mit dem ASN.1 Anzeigetool BERViewer
dargestellt und entsprechend kommentiert. Die Zertifizierungsantwort ist in mehrere Teile aufgegliedert, um eine bessere Übersicht zu erhalten.

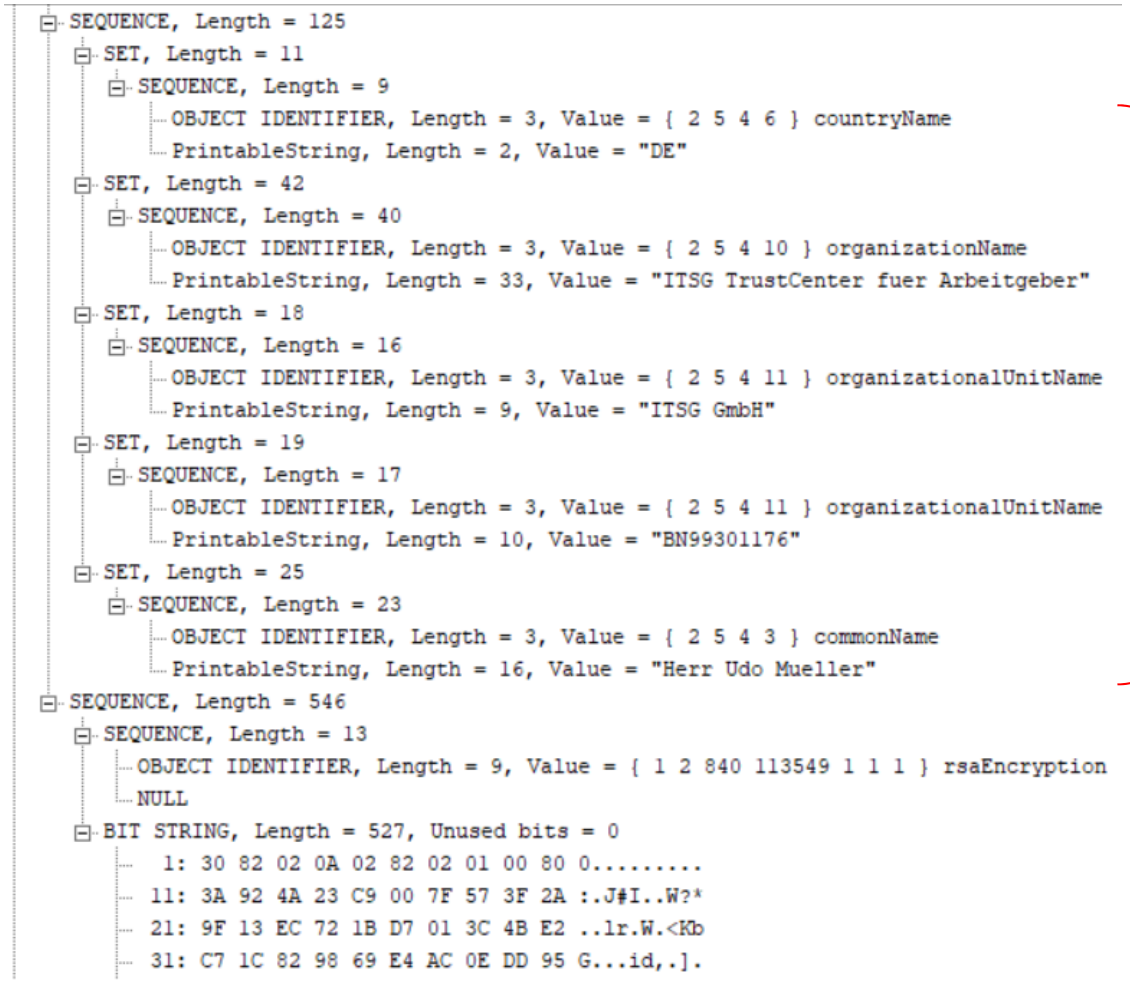
Übersicht einer Zertifizierungsantwort



Teil 1a


SEQUENCE, Length = 4419
OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 7 2 } signedData
[0] Context-Specific, Length = 4404
SEQUENCE, Length = 4400
INTEGER, Length = 1, Value = 1 (0x1)
SET, Length = 0
SEQUENCE, Length = 11
OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 7 1 } data
[0] Context-Specific, Length = 4374
SEQUENCE, Length = 1425
SEQUENCE, Length = 841
[0] Context-Specific, Length = 3
INTEGER, Length = 1, Value = 2 (0x2)
INTEGER, Length = 3, Value = 1000008 (0xF4248)
SEQUENCE, Length = 61
OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 10 } rsaPSS
SEQUENCE, Length = 48
[0] Context-Specific, Length = 13
SEQUENCE, Length = 11
OBJECT IDENTIFIER, Length = 9, Value = { 2 16 840 1 101 3 4 2 1 } sha-256
[1] Context-Specific, Length = 26
SEQUENCE, Length = 24
OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 8 } rsaOAEP-MGF
SEQUENCE, Length = 11
OBJECT IDENTIFIER, Length = 9, Value = { 2 16 840 1 101 3 4 2 1 } sha-256
[2] Context-Specific, Length = 3
INTEGER, Length = 1, Value = 32 (0x20)
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"
SEQUENCE, Length = 30
UTCTime, Length = 13, Value = "180410000000Z" (Apr 10, xx18 00:00:00 Zulu)
UTCTime, Length = 13, Value = "181130235959Z" (Nov 30, xx18 23:59:59 Zulu)
Zertifikat Serienummer, Kapitel 4.4.2
Gültig von, bis

Teil 1b



Distinguished Name,
Kapitel 4.4.4 und 4.4.5

Teil 1c

```
SEQUENCE, Length = 61
  OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 10 } rsaPSS
  SEQUENCE, Length = 48
    [0] Context-Specific, Length = 13
      SEQUENCE, Length = 11
        OBJECT IDENTIFIER, Length = 9, Value = { 2 16 840 1 101 3 4 2 1 } sha-256
    [1] Context-Specific, Length = 26
      SEQUENCE, Length = 24
        OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 8 } rsaOAEP-MGF
        SEQUENCE, Length = 11
          OBJECT IDENTIFIER, Length = 9, Value = { 2 16 840 1 101 3 4 2 1 } sha-256
    [2] Context-Specific, Length = 3
      INTEGER, Length = 1, Value = 32 (0x20)
BIT STRING, Length = 513, Unused bits = 0
   1: B7 80 C8 26 A0 A5 A0 E4 9B 9C 7.H& % d..
  11: E3 F3 B3 10 48 20 FF 1F 04 AD cs3.H ...-
  21: BB 1C F3 2F 75 D1 34 84 35 2E ;.s/uQ4.5.
  31: 67 F8 C6 4D 9F 92 E5 FC 8F 05 gxFM..e|..
```

Teil 2a

```
SEQUENCE, Length = 1525
  SEQUENCE, Length = 941
    [0] Context-Specific, Length = 3
      INTEGER, Length = 1, Value = 2 (0x2)
    INTEGER, Length = 1, Value = 71 (0x47)
    SEQUENCE, Length = 61
      OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 10 } rsaPSS
      SEQUENCE, Length = 48
        [0] Context-Specific, Length = 13
          SEQUENCE, Length = 11
            OBJECT IDENTIFIER, Length = 9, Value = { 2 16 840 1 101 3 4 2 1 } sha-256
        [1] Context-Specific, Length = 26
          SEQUENCE, Length = 24
            OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 8 } rsaOAEP-MGF
            SEQUENCE, Length = 11
              OBJECT IDENTIFIER, Length = 9, Value = { 2 16 840 1 101 3 4 2 1 } sha-256
        [2] Context-Specific, Length = 3
          INTEGER, Length = 1, Value = 32 (0x20)
    SEQUENCE, Length = 70
      SET, Length = 11
        SEQUENCE, Length = 9
          OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 6 } countryName
          PrintableString, Length = 2, Value = "DE"
      SET, Length = 55
        SEQUENCE, Length = 53
          OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 10 } organizationName
          PrintableString, Length = 46, Value = "Datenaustausch im Gesundheits- und Sozialwesen"
    SEQUENCE, Length = 30
      UTCTime, Length = 13, Value = "171218125835Z" (Dec 18, xx17 12:58:35 Zulu)
      UTCTime, Length = 13, Value = "2302152359595Z" (Feb 15, xx23 23:59:59 Zulu)
```

Teil 2b

```
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"
SEQUENCE, Length = 546
  SEQUENCE, Length = 13
    OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 1 } rsaEncryption
    NULL
  BIT STRING, Length = 527, Unused bits = 0
[3] Context-Specific, Length = 154
  SEQUENCE, Length = 151
    SEQUENCE, Length = 15
      OBJECT IDENTIFIER, Length = 3, Value = { 2 5 29 19 } basicConstraints
      OCTET STRING, Length = 8
        1: 30 06 01 01 FF 02 01 00       0.......
    SEQUENCE, Length = 11
      OBJECT IDENTIFIER, Length = 3, Value = { 2 5 29 15 } keyUsage
      OCTET STRING, Length = 4
        1: 03 02 01 06                   ....
    SEQUENCE, Length = 29
      OBJECT IDENTIFIER, Length = 3, Value = { 2 5 29 14 } subjectKeyIdentifier
      OCTET STRING, Length = 22
         1: 04 14 43 69 9D EF 73 78 BD 4E ..Ci.osx=N
        11: AE F9 94 21 43 65 F2 1A DE FE .y.!Cer.^~
        21: BA D3                         :S
    SEQUENCE, Length = 88
      OBJECT IDENTIFIER, Length = 3, Value = { 2 5 29 35 } authorityKeyIdentifier
      OCTET STRING, Length = 81
         1: 30 4F A1 4A A4 48 30 46 31 0B 0O!J$H0F1.
        11: 30 09 06 03 55 04 06 13 02 44 0...U....D
        21: 45 31 37 30 35 06 03 55 04 0A E1705..U..
```

Teil 2c

```
[3] Context-Specific, Length = 154
  SEQUENCE, Length = 151
    SEQUENCE, Length = 15
      OBJECT IDENTIFIER, Length = 3, Value = { 2 5 29 19 } basicConstraints
      OCTET STRING, Length = 8
        1: 30 06 01 01 FF 02 01 00       0.......
    SEQUENCE, Length = 11
      OBJECT IDENTIFIER, Length = 3, Value = { 2 5 29 15 } keyUsage
      OCTET STRING, Length = 4
        1: 03 02 01 06                   ....
    SEQUENCE, Length = 29
      OBJECT IDENTIFIER, Length = 3, Value = { 2 5 29 14 } subjectKeyIdentifier
      OCTET STRING, Length = 22
         1: 04 14 43 69 9D EF 73 78 BD 4E ..Ci.osx=N
        11: AE F9 94 21 43 65 F2 1A DE FE .y.!Cer.^~
        21: BA D3                         :S
    SEQUENCE, Length = 88
      OBJECT IDENTIFIER, Length = 3, Value = { 2 5 29 35 } authorityKeyIdentifier
      OCTET STRING, Length = 81
         1: 30 4F A1 4A A4 48 30 46 31 0B 0O!J$H0F1.
        11: 30 09 06 03 55 04 06 13 02 44 0...U....D
        21: 45 31 37 30 35 06 03 55 04 0A E1705..U..
        31: 13 2E 44 61 74 65 6E 61 75 73 ..Datenaus
        41: 74 61 75 73 63 68 20 69 6D 20 tausch im
        51: 47 65 73 75 6E 64 68 65 69 74 Gesundheit
        61: 73 2D 20 75 6E 64 20 53 6F 7A s- und Soz
        71: 69 61 6C 77 65 73 65 6E 82 01 ialwesen..
        81: 46                            F
```

Teil 3a

```
SEQUENCE, Length = 61
  OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 10 } rsaPSS
  SEQUENCE, Length = 48
    [0] Context-Specific, Length = 13
      SEQUENCE, Length = 11
        OBJECT IDENTIFIER, Length = 9, Value = { 2 16 840 1 101 3 4 2 1 } sha-256
    [1] Context-Specific, Length = 26
      SEQUENCE, Length = 24
        OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 8 } rsaOAEP-MGF
        SEQUENCE, Length = 11
          OBJECT IDENTIFIER, Length = 9, Value = { 2 16 840 1 101 3 4 2 1 } sha-256
    [2] Context-Specific, Length = 3
      INTEGER, Length = 1, Value = 32 (0x20)
BIT STRING, Length = 513, Unused bits = 0
     1: 57 AF 89 4B E5 1F D9 21 17 51 W/.Ke.Y!.Q
    11: 34 FE D7 BF DA A9 D4 5B EF E0 4~W?Z)T[o`
    21: C0 32 7F F3 51 E0 60 92 8D 74 @2.sQ``..t
    31: FA 21 C1 6E 3B 1A D0 8F 57 E5 z!An;.P.We
    41: E1 E4 71 EC AB B1 A1 C6 8E FA adql+1!F.z
    51: 71 40 24 44 9F 08 66 E6 5B 51 q@$D..ff[Q
    61: 04 C0 75 C5 35 25 FD BB 76 40 .@uE5%};v@
    71: 64 F9 FE 64 35 BA B6 C0 87 C1 dy~d5:6@.A
    81: F6 DA A0 45 2B 3B 20 0D 7F 95 vZ E+; ...
```

Teil 3b

```
SEQUENCE, Length = 1412
  SEQUENCE, Length = 828
    [0] Context-Specific, Length = 3
      INTEGER, Length = 1, Value = 2 (0x2)
    INTEGER, Length = 1, Value = 70 (0x46)
    SEQUENCE, Length = 61
      OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 10 } rsaPSS
      SEQUENCE, Length = 48
        [0] Context-Specific, Length = 13
          SEQUENCE, Length = 11
            OBJECT IDENTIFIER, Length = 9, Value = { 2 16 840 1 101 3 4 2 1 } sha-256
        [1] Context-Specific, Length = 26
          SEQUENCE, Length = 24
            OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 8 } rsaOAEP-MGF
            SEQUENCE, Length = 11
              OBJECT IDENTIFIER, Length = 9, Value = { 2 16 840 1 101 3 4 2 1 } sha-256
        [2] Context-Specific, Length = 3
          INTEGER, Length = 1, Value = 32 (0x20)
    SEQUENCE, Length = 70
      SET, Length = 11
        SEQUENCE, Length = 9
          OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 6 } countryName
          PrintableString, Length = 2, Value = "DE"
      SET, Length = 55
        SEQUENCE, Length = 53
          OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 10 } organizationName
          PrintableString, Length = 46, Value = "Datenaustausch im Gesundheits- und Sozialwesen"
    SEQUENCE, Length = 30
      UTCTime, Length = 13, Value = "1712141435222" (Dec 14, xx17 14:35:22 Zulu)
      UTCTime, Length = 13, Value = "2502141435222" (Feb 14, xx25 14:35:22 Zulu)
```

Teil 3c

```
SEQUENCE, Length = 70
  SET, Length = 11
  SET, Length = 55
    SEQUENCE, Length = 53
      OBJECT IDENTIFIER, Length = 3, Value = { 2 5 4 10 } organizationName
      PrintableString, Length = 46, Value = "Datenaustausch im Gesundheits- und Sozialwesen"
SEQUENCE, Length = 546
  SEQUENCE, Length = 13
    OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 1 } rsaEncryption
    NULL
  BIT STRING, Length = 527, Unused bits = 0
     1: 30 82 02 0A 02 82 02 01 00 CD 0........M
    11: A0 9D 22 E2 A1 F9 EA DD F6 67  ."b!yj]vg
    21: 68 2A 70 C5 1D 48 34 91 36 C1 h*pE.H4.6A
    31: E5 36 BE 2E 16 29 25 5E 6B 8A e6>..)%^k.
```

Teil 4

```
[3] Context-Specific, Length = 29
  SEQUENCE, Length = 27
    SEQUENCE, Length = 12
      OBJECT IDENTIFIER, Length = 3, Value = { 2 5 29 19 } basicConstraints
      OCTET STRING, Length = 5
        1: 30 03 01 01 FF                0....
    SEQUENCE, Length = 11
      OBJECT IDENTIFIER, Length = 3, Value = { 2 5 29 15 } keyUsage
      OCTET STRING, Length = 4
        1: 03 02 01 06                   ....
SEQUENCE, Length = 61
  OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 10 } rsaPSS
  SEQUENCE, Length = 48
    [0] Context-Specific, Length = 13
      SEQUENCE, Length = 11
        OBJECT IDENTIFIER, Length = 9, Value = { 2 16 840 1 101 3 4 2 1 } sha-256
    [1] Context-Specific, Length = 26
      SEQUENCE, Length = 24
        OBJECT IDENTIFIER, Length = 9, Value = { 1 2 840 113549 1 1 8 } rsaOAEP-MGF
        SEQUENCE, Length = 11
          OBJECT IDENTIFIER, Length = 9, Value = { 2 16 840 1 101 3 4 2 1 } sha-256
    [2] Context-Specific, Length = 3
      INTEGER, Length = 1, Value = 32 (0x20)
BIT STRING, Length = 513, Unused bits = 0
   1: 5D 59 B8 E3 CB 0E 7E 0E 1B 2A ]Y8cK.~..*
  11: 26 82 78 55 1A 41 2D 97 C5 47 &.xU.A-.EG
  21: C3 66 62 42 80 3B 7B D1 46 04 CfbB.;{QF.
```

Teil 5

```
[1] Context-Specific, Length = 0
SET, Length = 0
```

7. Literaturverzeichnis

7.1 Literaturverzeichnis

[ANS.1] X.208 CCITT Recommendation X.209: Specification of Abstract syntax Notation One (ASN.1), 1988
X.209 CCITT Recommendation X.209: Specification of basic ncoding rules for Abstract Syntax Notation One (ASN.1), 1988

[BNA-AlgKat] Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen), Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen, Entwurf vom 15.11.2016

[BSI-TR02102] Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Technische Richtlinie TR-02102-1, Bundesamt für Sicherheit in der Informationstechnik (BSI), Stand 08.02.2017 (Version 2017-01)

[EG-SIG]	Richtlinie 1999/93 EG des Europäischen Parlaments und Rates vom 13.Dezember 1999 über gemeinschaftliche Rahmenbedingungen für elektronische Signaturen
[ENISA-AlgRep]	Algorithms, Key Sizes and Parameters Report, 2013 recommendations, European Union Agency for Network and Information Security (ENISA), version 1.0 – October 2013
[FIPS180-3]	Federal Information Processing Standards (FIPS PUB) 180-3: Secure Hash Standard; October 2008
[CommonPKI-0]	Common PKI Specifications for Interoperable Applications; T7 & Tele-Trust; Introduction; V2.0 – 20 January 2009
[CommonPKI-1]	Common PKI Specifications for Interoperable Applications; T7 & Tele-Trust; Part 1: Certificate and CRL Profiles; V2.0 – 20 January 2009
[CommonPKI-2]	Common PKI Specifications for Interoperable Applications; T7 & Tele-Trust; Part 2: PKI Management; V2.0 – 20 January 2009
[CommonPKI-3]	Common PKI Specifications for Interoperable Applications; T7 & Tele-Trust; Part 3: CMS Bases Message Formats; V2.0 – 20 January 2009
[CommonPKI-4]	Common PKI Specifications for Interoperable Applications; T7 & Tele-Trust; Part 4: Operational Protocols; V2.0 – 20 January 2009
[CommonPKI-6]	Common PKI Specifications for Interoperable Applications; T7 & Tele-Trust; Part 6: Cryptographic Algorithms; V2.0 – 20 January 2009
[PKCS#1]	RSA Laboratories: PKCS #1 v.2.2: RSA Cryptography Standard, RSA Laboratories, 27 Oktober 2012; vgl. [RFC 8017]
[PKCS#7]	RSA Laboratories: PKCS #7: Cryptographic Message Syntax Standard; An RSA Laboratories Technical Note; Version 1.5; Revised November 1, 1993; vgl. [RFC 2315] bzw. [RFC 5652].
[PKCS#10]	RSA Laboratories: PKCS #10 v1.7: Certification Request Syntax Standard; May 26, 2000; vgl. [RFC 2314]
[PKCS#11]	RSA Laboratories. PKCS#11:Cryptographic Token Interface Standard, Version 2.3, 28.10.2009[RFC 4511] Lightweight Directory Access Protocol (LDAP): The Protocol. J. Sermersheim. June 2006
[RFC 22736299 98]	UTF-8, a transformation format of ISO 10646; F. Yergeau; Januar November 20031998
[RFC 2315]	PKCS #7: Cryptographic Message Syntax Version 1.5. B. Kaliski. March 1998. (Format: TXT=69679 bytes) (Status: INFORMATIONAL)
[RFC 5652]	Cryptographic Message Syntax, R. Housley, September 2009
[RFC 5754]	Using SHA2 Algorithms with Cryptograpfhic Message Syntax, S. Truner, January 2010
[RFC 8017]	PKCS #1: RSA Cryptography Specifications Version 2.2; K. Moriarty, B. Kaliski, J. Jonsson, A. Rusch; November 2016
[RFC 4513]	Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms. R. Harrison. June 2006

[RFC 2849] The LDAP Data Interchange Format (LDIF) - Technical Specification. G. Good. June 2000.

[RFC 3565] Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS), J. Schaad, July 2003

[RFC 4510] LDAP v2 Light-weight Directory Access Protocol

[RFC 5246] The Transport Layer Security (TLS) Protocol Version 1.2. Dierks & Rescorla. August 2008.[SecS2.0.1] Security Schnittstelle V2.0.1 – Stand Februar 2012

[X.500] CCITT: Recommendation X.500: The Directory Overview and Concepts, models and Services. 1988.

[X.509] CCITT: Recommendation X.509: The Directory-Authentication Framework. 1988.

7.2 Abkürzungsverzeichnis

CA	Certification Authority
CMS	Cryptographic Message Syntax
ISIS-MTT	Einheitlicher Interoperabilitäts-Standard der Trust Center ISIS (Industrial Signature Interoperability Specification)
MTRUST	TeleTrust: MailTrust Spezifikationen Version 2
LDAP	Lightweight Directory Access Protocol
OID	Object Identifier, Objektbezeichner
OCSP	Online Certificate Status Protocol
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