BSI

Bundesamt für Sicherheit in der Informationstechnik

SPEZIFIKATION ZUR ENTWICKLUNG INTEROPERABLER VERFAHREN UND KOMPONENTEN NACH SIGG/SIGV

SIGNATUR-INTEROPERABILITÄTSSPEZIFIKATION

SIGI

ABSCHNITT A2

SIGNATUR

STAND: 30. JUNI 1999

VERSION 6.1

Godesberger Allee 183, 53175 Bonn - Postfach 20 03 63, 53133 Bonn
Telefon: (0228) 9582 - 0, Telefax: (0228) 9582 - 400
Internet: www.bsi.bund.de

ABSCHNITT A2

SIGNATUR

Fritz Bauspieß, Jobst Biester, Dr. Dörte Neundorf

Secorvo Security Consulting GmbH
Albert-Nestler-Straße 9, D-76131 Karlsruhe

Tel. +49 721 6105-452
Fax +49 721 6105-455

E-Mail: info@secorvo.de
http://www.secorvo.de

Inhaltsverzeichnis

Abkürzungsverzeichnis

ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation 1
BER	Basic Encoding Rules
BGBI	Bundesgesetzblatt
BSI	Bundesamt für Sicherheit in der Informationstechnik
CER	Canonical Encoding Rules
CMS	Cryptographic Message Syntax
DER	Distinguished Encoding Rules
DIN	Deutsches Institut für Normung e.V.
DSA	Digital Signature Algorithm
DSI	Digital Signature Input
DSS	Digital Signature Standard
ECDSA	Elliptic Curve Digital Signature Algorithm
FDIS	Final Draft International Standard
FIPS	Federal Information Processing Standards
FIPS PUB	FIPS Publication
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
ITU	International Telecommunication Union
ITU-T	ITU - Telecommunication Sector
IuKDG	Informations- und Kommunikationsdienste-Gesetz
LNCS	Lecture Notes in Computer Science
MIME	Multipurpose Internet Mail Extension
MISPC	Minimum Interoperability Specification
NIST	National Institute of Standards and Technology
OID	Object Identifier
OIW	Open Systems Environment Implementors' Workshop
PKCS	Public-Key Cryptography Standard
PKI	Public Key Infrastructure
RACE	Research and Development in Advanced Communication Technologies in Europe
RegTP	Regulierungsbehörde für Telekommunikation und Post
RFC	Request for Comments
RIPE	RACE Integrity Primitives Evaluation
RIPEMD	RIPE Message Digest
RSA	Rivest, Shamir, Adleman Kryptosystem
S/MIME	Secure MIME
SET	Secure Electronic Transactions
SHA	Secure Hash Algorithm

SHS	Secure Hash Standard
SigG	Signaturgesetz
SigI	Signatur-Interoperabilitätsspezifikation
SigV	Signaturverordnung
TTT	TeleTrusT Deutschland e.V.
ZS	Zertifizierungsstelle

1 Inhalt des Dokuments

Das vorliegende Dokument besteht aus zwei Teilen, die einen unterschiedlichen Anwendungsbereich haben. Der erste Teil enthält die genauen Festlegungen der im Rahmen von SigI zu verwendenden Algorithmen für alle zulässigen Signaturverfahren. Der zweite Teil beschreibt Signatur- und Austauschformate für Nachrichten[1].

Die Festlegungen des ersten Teils gelten nicht nur für die im zweiten Teil beschriebenen Nachrichten, sondern allgemein für alle Arten digital signierter Daten. Die zulässigen Signaturverfahren sind für alle diese Arten gleich.

2 Zusammenfassung

Im Kapitel 3 „Signaturverfahren“ werden Festlegungen für die zur Signaturerzeugung und -verifikation zulässigen Algorithmen auf der Basis von [BAZ140298] getroffen. Zulässige Algorithmen sind:

- Die Hashalgorithmen SHA-1 und RIPEMD-160
- Der auf dem Problem der Faktorisierung großer Zahlen basierende RSA-Algorithmus mit zwei verschiedenen Verfahren zur Formatierung des Eingabewertes für die Bildung der digitalen Signatur
- Der auf dem diskreten Logarithmusproblem basierende DSA-Algorithmus
- Eine auf elliptischen Kurven basierende Erweiterung des DSA (ECDSA).

Im Kapitel 4 „Signaturumfang“ wird beschrieben, welche Daten zusammen mit einer digitalen Signatur in Nachrichten ausgetauscht werden müssen (obligatorische Daten). Es werden weitere Daten identifiziert, die der Signierer der Signatur beifügen kann (optionale Daten).[2] Alle Daten werden danach unterschieden, ob sie in die digitale Signatur einzubeziehen sind oder nicht. Durch die Menge der in die Signatur einzubeziehenden obligatorischen und der optionalen Daten wird der Signaturumfang bestimmt.

In den Kapiteln 5 „Signaturaustauschformat“, 6 „Attributtypen“ und 7 „Signaturbildung“ wird das Format der zusammen mit der Signatur auszutauschenden Daten unter Verwendung der Spezifikationssprache ASN.1 beschrieben. Das Signaturaustauschformat orientiert sich an der Spezifikation des Datentyps „SignedData“, der in der Cryptographic Message Syntax (CMS) der IETF enthalten ist.

Das CMS-Format bildet die Basis für das hier spezifizierte Signaturgesetz-konforme Signaturaustauschformat. Es wurde um zusätzliche Attribute ergänzt, um den Anforderungen des Signaturgesetzes gerecht zu werden. Andere Attribute wurden ergänzt, um Anforderungen der Nutzer besser entsprechen zu können und damit die Benutzerfreundlichkeit zu erhöhen.

Hersteller Signaturgesetz-konformer Applikationen können der Spezifikation entnehmen, welche Teile sie realisieren müssen und welche Teile optional sind. Es wird stets angegeben, ob die hier getroffenen Festlegungen aus Anforderungen des Gesetzes oder der Verordnung abgeleitet sind, oder ob sie aus anderen Gründen in die Spezifikation aufgenommen wurden.

---

1 Unter einer Nachricht wird hier eine bestimmte Art digitaler Signatur verstanden. Nachrichten sind digitale signierte Daten, die zwischen Teilnehmern ausgetauscht werden. Diese Definition ermöglicht eine einfache Unterscheidung zu anderen digital signierten Daten wie Zertifikate, Sperrlisten, Auskünfte des Verzeichnisdienstes, etc.

2 Optionale Daten können z.B. beigefügt werden, um dem Verifizierer die Prüfung der digitalen Signatur zu erleichtern oder ihm ergänzende Informationen zur Signatur zur Verfügung zu stellen.

Bei optionalen Teilen der Spezifikation können die Hersteller frei entscheiden, ob sie diese implementieren. Falls optionale Teile der Spezifikation implementiert werden, so muß die Implementierung jedoch so erfolgen, wie hier spezifiziert, um auch für diesen Bereich Interoperabilität herstellen zu können - zumindest zwischen den Implementierungen, die diese optionalen Anteile unterstützen.

Aus Gründen der Interoperabilität und um den Implementierungsaufwand für Hersteller zu verringern wurde die Vielzahl der Optionen der CMS soweit als möglich reduziert, ohne daß dadurch Einbußen bei der Funktionalität zu erwarten sind.

Kapitel 8 „Austauschformat“ enthält ein Format für den Austausch von Nachrichten, die durch eine digitale Signatur gesichert sind. Die digitalen Daten werden so in eine Nachricht eingebettet, daß sie auf dem Transportweg nicht unbemerkt verändert und vom Empfänger automatisch ausgewertet werden können. Das hier spezifizierte Austauschformat basiert auf S/MIME, wobei zur Reduktion der Komplexität von S/MIME und zur Erfüllung der SigG-spezifischen Anforderungen hier ein SigI-spezifisches Profile für S/MIME definiert ist. Die Unterstützung dieses Austauschformats ist nach dieser Spezifikation nicht obligatorisch, wird jedoch empfohlen.

3 Signaturverfahren

Eine **Signatur** ist in § 2 Abs. 1 SigG als ein mit einem privaten Signaturschlüssel erzeugtes Siegel zu **digitalen Daten** definiert. Die Signatur hat den Zweck, für jeden Verifizierer der Signatur den Inhaber des Signaturschlüssels und die Unverfälschtheit der digitalen Daten erkennbar zu machen. Sie dient somit der Sicherung der Echtheit und Unverfälschtheit der Daten.

Signaturen werden in der Regel dazu verwendet, um den Empfänger digitaler Daten davon zu überzeugen, daß die Daten vom **Signaturschlüssel-Inhaber** als Ersteller der Signatur stammen und nicht verfälscht wurden.[3] Der Empfänger oder jeder beliebige Dritte kann dies durch Prüfung der Signatur jederzeit feststellen. Falls die digitalen Daten als Willenserklärung interpretiert werden können, dient ein positives Ergebnis der Prüfung der Signatur als Beweismittel für deren rechtliche Wirksamkeit.

Im Rahmen von SigI werden verschiedene Arten digital signierter Daten unterschieden. Digitale Signaturen werden sowohl zum gesicherten Austausch von Informationen zwischen zwei Teilnehmern der Sicherheitsinfrastruktur als auch zwischen Zertifizierungsstellen und Teilnehmern verwendet.

Die Verfahren zur Erzeugung und Verifikation digitaler Signaturen sind in allen Fällen gleich. Unterschiedlich sind jedoch die Formate, die verwendet werden, um digitale Signaturen austauschen zu können. Die Formate sind Gegenstand der weiteren Kapitel des vorliegenden Dokumentes soweit der Austausch von Nachrichten betroffen ist. Formate zum Austausch anderer Arten digital signierter Daten sind in anderen Dokumenten der Gesamtspezifikation enthalten.[4]

Jedes **Signaturverfahren** bzw. Signaturschema besteht aus einem Algorithmus zur Erzeugung digitaler Signaturen und einem zugeordneten Algorithmus zur Verifikation digitaler Signaturen. Für SigI sind verschiedene Signaturverfahren zugelassen. Als Basis dient die Veröffentlichung

---

3 Allgemein dienen Signaturen zur sicheren Übertragung in Raum (sichere Kommunikation) und Zeit (sichere Speicherung / Archivierung).

4 Zertifikate sind in [BSI-ZERT 99], Verzeichnisdienstauskünfte in [BSI-DIR 99] und Zeitstempel in [BSI-TSP 99] spezifiziert.

der zuständigen Behörde gemäß § 17 Abs. 2 SigV im Bundesanzeiger [BAZ140298], in der geeignete Algorithmen genannt sind.[5]

Allen zugelassenen Signaturverfahren ist gemeinsam, daß eine sogenannte Signatur mit Anhang (signature with appendix) gebildet wird. Die digitalen Daten, für die eine digitale Signatur gebildet wird, werden dabei als Anhang den digital signierten Daten beigefügt.

Jeder Algorithmus zur Erzeugung digitaler Daten erhält als Input (DSI) die zu sigierenden Daten, den privaten Schlüssel des Signieres sowie weitere Parameter und liefert als Output die digitale Signatur. Der zugeordnete Algorithmus zur Verifikation digitaler Signaturen erhält als Input digital signierte Daten, den öffentlichen Schlüssel des Signierers sowie weitere Parameter und liefert als Ergebnis „mathematisch korrekt“, falls die Verifikation erfolgreich war.[6]

Die Signaturbildung erfolgt stets in folgenden Schritten:

- Bestimmung des Hashwertes der digitalen Daten
- Ggf. Anwendung einer Kodierungsoperation auf den Hashwert (Padding)[7]
- Berechnung der digitalen Signatur durch Anwendung der Signaturfunktion
- Ausgabe der digitalen Signatur

Die Verifikation digitaler Signaturen erfolgt in folgenden Schritten:

- Bestimmung des Hashwertes der digitalen Daten aus dem Anhang zur digitalen Signatur
- Ggf. Anwendung einer Kodierungsoperation auf den Hashwert
- Anwendung einer Verifikationsfunktion auf die digitale Signatur und den Hashwert
- Ausgabe des Verifikationsergebnisses

Die zulässigen Signaturverfahren unterscheiden sich durch die verwendete Signatur- und Verifikationsfunktion (RSA, DSA oder ECDSA), den verwendeten Hashalgorithmus zur Bestimmung des Hashwertes (SHA-1 oder RIPEMD-160) und das verwendete Paddingverfahren (bei RSA). Beide Hashalgorithmen können zusammen mit jeder der Signatur- und Verifikationsfunktionen verwendet werden. Das Padding ist dagegen nur bei RSA von Bedeutung.

Achtung: Durch die Zulassung der Kombination beider Hash-Algorithmen mit allen zugelassenen Signatur- und Verifikationsfunktionen werden bestehende Standards erweitert. Die Standards für DSA und ECDSA sehen vor, daß diese nur mit SHA-1 verwendet werden dürfen. Die Erweiterung erlaubt auch die Verwendung von RIPEMD-160. Dadurch wird sichergestellt, daß die Verwendbarkeit einer Signatur- und Verifikationsfunktion nicht von der Güte eines einzigen Hashalgorithmus abhängt.[8]

---

5 Alle für SigI zugelassenen Signaturverfahren werden von der zuständigen Behörde als geeignet angesehen.

6 Für eine differenziertere Betrachtung siehe Kapitel 3.4 [BSI-GM 99]

7 Die Aufgabe des Paddings ist es, den Hashwert um einen sogenannten Padding-String zu ergänzen, falls die Länge des Hashwertes kürzer ist als die erforderliche Länge des DSI.

8 Jede Signatur- und Verifikationsfunktion soll mit einer anderen Hashalgorithmus auch dann noch verwendet werden können, falls sich später herausstellen sollte, daß eine der Hashalgorithmen als „schwach“ angesehen werden muß. Auch wenn dies nicht sehr wahrscheinlich ist, so soll angesichts der andernfalls möglicherweise gravierenden Auswirkungen aus Gründen der Vorsorge stets eine Alternative bestehen.

Aus Gründen der Interoperabilität muß sichergestellt sein, daß jeder Empfänger einer SigI-konformen Signatur technisch in der Lage ist, diese zu verifizieren. Deshalb müssen die zugelassenen Signaturverfahren danach unterschieden werden, ob sie bei der Generierung von Signaturen allgemein verwendet werden dürfen oder nicht. Signaturverfahren dürfen nur dann allgemein verwendet werden, wenn die vorliegende Spezifikation fordert, daß sie auf Empfängerseite verifiziert werden können. Die Generierung von Signaturen unter Anwendung zwar zugelassener aber nicht allgemein verwendbarer Signaturverfahren darf aus Gründen der Interoperabilität nur dann erfolgen, wenn auf andere nicht weiter spezifizierte Weise sichergestellt ist, daß die Signaturen von allen bestimmungsgemäßen Empfängern verifiziert werden können.

Die vorliegende Spezifikation berücksichtigt ein Migrationskonzept für ECDSA. Während RSA und DSA von allen Verifikationskomponenten SigI-konformer Produkte unterstützt werden müssen ist dies für ECDSA erst mittelfristig vorgesehen. Dadurch wird darauf Rücksicht genommen, daß Implementierungen von ECDSA noch nicht sehr verbreitet sind.[9]

Für die Komponenten zur Signaturbildung (Signierkomponenten) gilt:

- Es muß entweder RSA oder DSA unterstützt werden
- Es besteht die freie Auswahl zwischen RSA und DSA bei der Generierung von Signaturen
- ECDSA darf bei der Generierung von Signaturen nur dann verwendet werden, wenn sichergestellt ist, daß die Signaturen von den bestimmungsgemäßen Empfängern verifiziert werden können

Für die Komponenten zur Verifikation digitaler Signaturen (Prüfkomponenten) gilt:

- Die Verfahren RSA und DSA müssen unterstützt werden
- Die Unterstützung von ECDSA ist zunächst optional

Im folgenden Kapitel 3.1 werden die beiden Hashalgorithmen vorab beschrieben, da sie in Kombination mit allen Signatur- und Verifikationsfunktion verwendet werden können. In Kapitel 3.2 werden dann alle zugelassenen Signaturverfahren inklusive Padding beschrieben.

Signaturverfahren müssen in digital signierten Daten stets eindeutig gekennzeichnet werden, damit die Prüfkomponente erkennen kann, wie die Prüfung zu erfolgen hat. Signaturverfahren werden über Object Identifier (OIDs) gekennzeichnet. Dabei ist zu berücksichtigen, daß sowohl OIDs zur Identifizierung des Verfahrens inklusive Hashalgorithmus als auch zur Identifizierung des Verfahrens exklusive Hashalgorithmus definiert sind. Bei gleichem Signaturverfahren werden verschiedene OIDs für verschiedene Zwecke verwendet.

OIDs werden daneben auch zur Identifizierung von Hashalgorithmen verwendet. Die Verwendung aller Algorithmen-Identifizierer wird in Kapitel 3.3 dargestellt. Durch die genannten Erweiterungen der Standards und Regelungen in der Chipkartenspezifikation des DIN [DINChip 98] wurden hinsichtlich der Verwendung der OIDs Abweichungen von Standards erforderlich, auf denen SigI basiert. Diese Abweichungen werden begründet.

Alle OIDs werden anschließend in Kapitel 3.4 zusammengestellt.

3.1 Hashalgorithmen

Kryptographische Hashalgorithmen werden zur Bildung eines Hashwertes als erster Schritt bei der Generierung digitaler Signaturen und deren Verifikation verwendet. Sie erzeugen aus einer Eingabe variabler Länge eine Ausgabe fester Länge (Hashwert) und müssen die Eigenschaft

---

[9] Hersteller sollen nicht gezwungen sein, ECDSA kurzfristig zu implementieren. Mittelfristig müssen ihre Produkte jedoch auch ECDSA beherrschen.

besitzen, kollisionsresistent zu sein. Das bedeutet, daß es mit realistischem Aufwand nicht möglich sein darf, zwei verschiedene Eingabedaten zu finden, die auf denselben Hashwert abgebildet werden.

Für SigI sind die Hashalgorithmen SHA-1 und RIPEMD-160 zugelassen.[10] Zu jeder Hashfunktion werden folgende Angaben gemacht:

- zu verwendender Object Identifier
- Parameter des Algorithmus[11]
- Referenz auf den Standard, in dem der Algorithmus spezifiziert ist
- Kurzbeschreibung der Algorithmus

3.1.1 SHA-1

- **Object Identifier:**

  sha-1 OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 26 }[12]

- **Parameter:**

  SigI-konforme Komponenten **sollen** bei einer Angabe des „AlgorithmIdentifier“ für SHA-1 den ASN.1-Datentyp NULL als Parameter verwenden. Komponenten **müssen** bei der Auswertung sowohl die Angabe NULL im Feld „parameters“ als auch das Fehlen des Feldes „parameters“ korrekt interpretieren können.[13]

- **Referenz:**

  SHA-1 ist in [FIPS180-1 95] spezifiziert.

- **Kurzbeschreibung:**

  SHA-1 erhält als Eingabe einen Bitstring praktisch beliebiger Länge[14] und liefert als Ausgabe einen Bitstring der Länge 160.

3.1.2 RIPEMD-160

- **Object Identifier**:

---

10 Eine Übersicht der geeigneten Krypto-Algorithmen ist im Bundesanzeiger veröffentlicht (s. § 17 Abs. 2 SigV). Die Hashfunktionen SHA-1 und RIPEMD-160 werden z.Zt. bis Ende 2003 als geeignet angesehen (s. [BAZ140298]).

11 Unter "Parameter" werden hier die Steuergrößen des Algorithmus verstanden, die im Feld „parameters“ der Datenstruktur „AlgorithmIdentifier“ angegeben werden müssen (s. Kapitel 3.3).

Die Information, daß für den Algorithmus kein Parameter angegeben wird, kann prinzipiell auf zwei verschiedene Arten kodiert werden. Eine Möglichkeit besteht darin, das optionale Feld „parameters“ auszulassen. Die andere Möglichkeit besteht darin, den ASN.1-Datentyp NULL zu verwenden.

12 Dieser OID wird in allen Standards verwendet, auf denen SigI basiert.

13 Dies entspricht im Wesentlichen den Vorgaben in Kapitel 12.1.1 [CMS 98]. Die Vorgaben wurden verschärft (müssen), um Interoperabilitätsprobleme zu vermeiden.

14 Ein Bitstring ist eine Folge von Bits beliebiger Länge. Die Beschränkung der Eingabe auf Werte mit einer Länge kleiner als 2^{64} ist in der Praxis ohne Bedeutung.

ripemd-160 OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) teletrust(36) algorithm(3) hashAlgorithm(2) 1 }

- **Parameter:**

  SigI-konforme Komponenten **sollen** bei einer Angabe des „AlgorithmIdentifier“ für RIPEMD-160 den ASN.1-Datentyp NULL als Parameter verwenden. Komponenten **müssen** bei der Auswertung sowohl die Angabe NULL im Feld „parameters“ als auch das Fehlen des Feldes „parameters“ korrekt interpretieren können.[15]

- **Referenz:**

  RIPEMD-160 ist in [RIPEMD-160 96] und [ISO10118-3 98] spezifiziert.

- **Kurzbeschreibung:**

  RIPEMD-160 erhält als Eingabe einen Bitstring praktisch beliebiger Länge[16] und liefert als Ausgabe einen Bitstring der Länge 160.

3.2 Signaturalgorithmen

Die vorliegende Spezifikation umfaßt die drei Algorithmen RSA, DSA und ECDSA. Zu diesen Algorithmen werden jeweils folgende Angaben gemacht:

- zu verwendende Object Identifier
- Parameter des Algorithmus[17]
- allgemeine Referenz auf den Standard, in dem der Algorithmus spezifiziert ist
- Referenz auf die grundlegende mathematische Operation[18] für die Signaturbildung (Signaturfunktion)
- Referenz auf die grundlegende mathematische Operation für die Verifikation (Verifikationsfunktion)
- Konvertierungen von Daten zwischen verschiedenen Formaten
- Paddingverfahren
- Sicherheitsanforderungen[19]

---

15 Da keine Vorgaben für RIPEMD-160 existieren, wurden die Vorgaben für SHA-1 übernommen.

16 Ein Bitstring ist eine Folge von Bits beliebiger Länge. Die Beschränkung der Eingabe auf Werte mit einer Länge kleiner als 2^{64} ist in der Praxis ohne Bedeutung.

17 Unter "Parameter" werden hier die Steuergrößen des Algorithmus verstanden, die im Feld „parameters“ der Datenstruktur „AlgorithmIdentifier“ angegeben werden müssen.

Die Information, daß für den Algorithmus kein Parameter angegeben wird, kann prinzipiell auf zwei verschiedene Arten kodiert werden. Eine Möglichkeit besteht darin, das optionale Feld „parameters“ auszulassen. Die andere Möglichkeit besteht darin, den Datentyp NULL (ASN.1-Notation: NULL) zu verwenden.

18 Diese mathematischen Operationen werden auch Primitive (primitives) genannt.

19 Sicherheitsanforderungen werden hier nur genannt, soweit sie für die Interoperabilität relevant sind.

3.2.1 RSA

Die vorliegende Spezifikation unterscheidet vier Signaturverfahren auf der Basis von RSA, die sich durch die Wahl des Hashalgorithmus und das Padding unterscheiden.

Die Erzeugung einer digitalen Signatur erfolgt stets in folgenden Schritten:

- Berechnung des Hashwertes der zu signierenden Daten mittels eines der in Kapitel 3.1 beschriebenen Verfahren
- Anwendung einer Kodierungsoperation auf den Hashwert (s.u.)
- Konvertierung aus einer Kodierung als Oktet-String in eine Integerdarstellung
- Anwendung der Signaturfunktion mit dem privaten Schlüssel
- Konvertierung des erhaltenen Integerwertes in einen Oktet-String/Bitstring
- Ausgabe der digitalen Signatur

Die Verifikation einer digitalen Signatur erfolgt stets in folgenden Schritten:

- Konvertierung der digitalen Signatur in eine Integerdarstellung
- Anwendung der Verifikationsfunktion mit dem öffentlichen Schlüssel
- Konvertierung in eine Darstellung als Oktet-String (Zwischenergebnis)
- Bestimmung des Hashwertes der zu verifizierenden Daten mittels des bei der Erzeugung der Signatur verwendeten Hashverfahrens
- Anwendung einer Kodierungsoperation auf den Hashwert
- Vergleich des kodierten Hashwertes mit dem Zwischenergebnis
- Ausgabe des Verifikationsergebnisses

Die Aufgabe des Kodierungsverfahrens ist, den Hashwert mit einer Länge von 160 Bit um einen sogenannten Padding-String zu ergänzen, so daß nach der Kodierung insgesamt die vorgegebene Schlüssellänge von mindestens 1024 Bit erreicht wird. Die folgenden zwei Kodierungsverfahren sind zulässig:

- RSA mit Padding gemäß Kapitel 9.2 [PKCS1 98] (PKCS-Padding)
- RSA mit Padding gemäß Kapitel 2.1.1 [DINChip 98] (ISO9796-2rnd-Padding)[20]

3.2.1.1 Object Identifier

Für RSA werden insgesamt sieben OIDs benötigt.

Folgende registrierte OIDs sind zu verwenden:

- RSA mit PKCS-Padding:

  rsaEncryption OBJECT IDENTIFIER ::= { iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1 }

- RSA mit PKCS-Padding und SHA-1:

  sha1WithRSAEncryption OBJECT IDENTIFIER ::= { iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) pkcs-1(1) 5 }

---

[20] Das Padding ist eine Variante des in Kapitel 6 [ISO 9796-2 97] beschriebenen Paddingverfahrens.

- RSA mit PKCS-Padding und RIPEMD-160:

  rsaSignatureWithripemd160 OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) rsaSignature(1) 2 }

Die OIDs für RSA mit ISO9796-2rnd-Padding müssen noch registriert werden. Es werden die folgenden symbolischen OIDs verwendet[21]:

- rsa-padding-iso9796-2rnd-indicate-sha1 für RSA und SHA-1
- rsa-padding-iso9796-2rnd-indicate-ripemd160 für RSA und RIPEMD-160
- rsa-padding-iso9796-2rnd-with-sha1 für RSA und SHA-1
- rsa-padding-iso9796-2rnd-with-ripemd160 für RSA und RIPEMD-160

3.2.1.2 Parameter

SigI-konforme Komponenten **müssen** bei einer Angabe des „AlgorithmIdentifier“ für den RSA-Algorithmus im Feld „parameters“ stets den ASN.1-Wert NULL eintragen.[22]

3.2.1.3 Referenz

Der RSA-Algorithmus ist in [PKCS1 98] beschrieben.

3.2.1.4 Signaturfunktion

Der Algorithmus ist in Kapitel 5.2.1 [PKCS1 98] beschrieben.

3.2.1.5 Verifikationsfunktion

Der Algorithmus ist in Kapitel 5.2.2 [PKCS1 98] beschrieben.

3.2.1.6 Konvertierungen

Die folgenden Konvertierungen zwischen den ASN.1-Datentypen Integer und Oktet-String sind in Kapitel 4 [PKCS1 98] beschrieben:

- I2OSP Konvertierung eines Integerwertes in einen Oktet-String
- OS2IP Konvertierung eines Oktet-Strings in einen Integerwert

Ein Oktet-String ist eine geordnete Folge von Oktets, wobei das erste Oktet am weitesten links, das letzte am weitesten rechts angeordnet ist. Bei der Konvertierung in oder von einem Integerwert wird das erste Oktet als das mit der höchsten Signifikanz angesehen.

Neben diesen Konvertierungen werden auch Konvertierungen in oder von einem Bitstring in einen Integerwert benötigt. Digitale Signaturen werden entweder als Oktet-String oder als Bitstring dargestellt. Die digitale Signatur eines Zertifikates ist z.B. ein Bitstring (vgl. Kapitel 2.2 [BSI-ZERT 99], während die digitale Signatur einer Nachricht ein Oktet-String ist (vgl. Kapitel 5.5).

---

21 Eine Begründung für die Notwendigkeit der Registrierung dieser OIDs wird in Kapitel 3.3 gegeben.

22 Dies entspricht den Vorgaben in Kapitel 11 [PKCS1 98] (shall) und Kapitel 12.2.2 [CMS 98] (must).

Eine Konvertierung eines Intergerwertes in einen Bitstring erfolgt im Allgemeinen in zwei Schritten. Der Integerwert wird zunächst in einen Oktet-String und dieser dann in einen Bitstring konvertiert. Es werden daher die folgenden Konvertierungen benötigt, die in Kapitel 5.5.2 [IEEE P1363 98] beschrieben sind:

- BT2OSP Konvertierung eines Bitstrings in einen Oktet-String
- OS2BTP Konvertierung eines Oktet-Strings in einen Bitstring

3.2.1.7 Padding

Es werden die folgenden Padding-Varianten spezifiziert (s.o.):

- PKCS-Padding gemäß Kapitel 9.2 [PKCS1 98]
- ISO9796-2rnd-Padding gemäß Kapitel 2.1.1 [DINChip 98]

3.2.1.8 Sicherheitsanforderungen

Bei der Anwendung des RSA-Algorithmus muß der Modulus n = p q (p und q Primzahlen) eine Länge von mindestens 1024 Bit haben. Als maximale Länge des Moduls wird im Rahmen dieser Spezifikation aus Interoperabilitätsgründen 4096 Bit vorgegeben. Verifikationskomponenten müssen somit digitale Signaturen mit einer Länge von bis zu 4096 Bit prüfen können.

3.2.2 DSA

Die vorliegende Spezifikation umfaßt zwei Signaturverfahren auf der Basis von DSA, die sich durch die Wahl des Hashalgorithmus unterscheiden.

Die Erzeugung einer digitalen Signatur erfolgt stets in folgenden Schritten:

- Bestimmung des Hashwertes der zu signierenden Daten mittels eines der in Kapitel 3.1 beschriebenen Verfahren
- Konvertierung aus einer Kodierung als Bitstring in eine Integerdarstellung
- Anwendung der Signaturfunktion mit dem privaten Schlüssel und den DSA-Parametern
- Konvertierung der erhaltenen Integerwerte (die Signatur besteht aus den Werten r und s) in einen Oktet-String/Bitstring
- Ausgabe der digitalen Signatur

Die Verifikation einer digitalen Signatur erfolgt stets in folgenden Schritten:

- Bestimmung des Hashwertes der zu verifizierenden Daten mittels des bei der Erzeugung der Signatur verwendeten Hashalgorithmus
- Konvertierung aus einer Kodierung als Bitstring in eine Integerdarstellung
- Konvertierung der digitalen Signatur, d.h. der Werte r und s, in eine Integerdarstellung
- Anwendung der Verifikationsfunktion mit dem öffentlichen Schlüssel
- Ausgabe des Verifikationsergebnisses

3.2.2.1 Object Identifier

Für DSA werden insgesamt vier OIDs benötigt.

Folgende registrierte OIDs sind zu verwenden:

- DSA:

  id-dsa OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) x9-57(10040)x9cm(4) 1 }[23]

- DSA mit SHA-1:

  id-dsa-with-sha1 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) x9-57(10040)x9cm(4) 3 }[24]

Folgende symbolische OIDs müssen noch registriert werden:

- dsa-extended[25]
- dsa-with-ripemd160

3.2.2.2 Parameter

Der DSA-Algorithmus verwendet die drei allgemeinen Parameter p, q, g (vgl. Kapitel 4 [FIPS186-1 98]). Diese Parameter können öffentlich bekannt sein. Die gleichen Parameter können mehreren Schlüsselpaaren zugeordnet sein.

Die Parameter müssen dem Verifizierer bei der Prüfung digitaler Signaturen bekannt sein. Sie müssen im Teilfeld „parameters“ des Feldes „AlgorithmIdentifier“ der Datenstruktur SubjectPublicKeyInfo des Signaturschlüssel-Zertifikats[26] des Signierers enthalten sein, sofern sie nicht „vererbt“ werden.

Zertifizierungsstellen können die Parameter ihres eigenen Signaturschlüssel-Zertifikats optional vererben. Sofern sie Zertifikate für öffentliche Schlüssel ausstellen, deren Parameter mit ihren eigenen Parametern übereinstimmen, müssen in diesen Zertifikaten die Parameter nicht angegeben werden (vgl. [MISPC1 97]). Der Verifizierer muß die Parameter in diesem Fall aus dem übergeordneten Zertifikat der Zertifizierungsstelle entnehmen.

Auch die Wurzel-Zertifizierungsstelle kann ihre Parameter an die Zertifizierungsstellen vererben, so daß in diesem Fall auch in den Zertifikaten der Zertifizierungsstellen keine Parameter enthalten sind. Eine Vererbung ist somit auch über mehrere Stufen möglich. Die Vererbung muß von allen Komponenten unterstützt werden.

Achtung: Die Forderung nach Unterstützung der Vererbung bedeutet eine Ergänzung der Vorgaben des DSA-Standards. Der Standard sieht vor, daß Parameter auf externem Wege bereitgestellt werden, regelt aber nicht, wie dies geschehen soll.

SigI-konforme Komponenten **müssen** das Feld „parameters“ stets leer lassen, wenn keine Parameter angegeben werden.[27]

---

23 Vgl. Annex C1 [ANSI X9.57 97].

24 Vgl. Annex C1 [ANSI X9.57 97].

25 Der DSA-Standard erlaubt nur die Verwendung von SHA-1. Für die Verwendung von RIPEMD-160 muß der Anwendungsbereich von DSA erweitert werden. Eine Begründung für die Notwendigkeit eines eigenen OIDs wird in Kapitel 3.3 gegeben.

26 Vgl. Kapitel 2.3.7 [BSI-ZERT 99].

27 Dies entspricht den Vorgaben in Kapitel 5.2 [ANSI X9.57 97] (is omittet) und Kapitel 12.2.1 [CMS 98] (must not).

3.2.2.3 Referenz

Der DSA ist in [FIPS186-1 98] spezifiziert.

3.2.2.4 Signaturfunktion

Der Algorithmus ist in Kapitel 5 [FIPS186-1 98] beschrieben.

3.2.2.5 Verifikationsfunktion

Der Algorithmus ist in Kapitel 6 [FIPS186-1 98] beschrieben.

3.2.2.6 Konvertierungen

Siehe Kapitel 3.2.1.6.

3.2.2.7 Sicherheitsanforderungen

Während p gemäß Kapitel 4 [FIPS186-1 98] eine Länge von höchstens 1024 Bit haben darf wird in [BAZ140298] gefordert, daß die Länge mindestens 1024 Bit betragen muß.

Achtung: Der DSA-Standard wird im Rahmen von SigI hinsichtlich der Länge von p verschärft.

Als maximale Länge von p wird im Rahmen dieser Spezifikation aus Interoperabilitätsgründen 4096 Bit vorgegeben.

Zertifizierungstellen dürfen daher nur Zertifikate für Schlüssel ausstellen, falls die Länge von p mindestens 1024 und höchstens 4096 Bit beträgt. Signatur- und Verifikationskomponenten müssen Werte für p mit einer Länge von bis zu 4096 Bit verarbeiten können.

Während q gemäß Kapitel 4 [FIPS186-1 98] eine Länge von exakt 160 Bit haben muß wird in [BAZ140298] gefordert, daß die Länge mindestens 160 Bit betragen muß.

Achtung: Der DSA-Standard wird im Rahmen von SigI hinsichtlich der Länge von q verschärft.

Als maximale Länge von q wird im Rahmen dieser Spezifikation aus Interoperabilitätsgründen 320 Bit vorgegeben.

Zertifizierungstellen dürfen daher nur Zertifikate für Schlüssel ausstellen, falls die Länge von q mindestens 160 und höchstens 320 Bit beträgt. Signatur- und Verifikationskomponenten müssen Werte für q mit einer Länge von bis zu 320 Bit verarbeiten können.

Da die Verschärfung der Sicherheitsanforderungen durch die Erhöhung der Mindestlängen für p und q nicht zu Sicherheitsproblemen führen kann, können die Standard-OIDs für DSA beibehalten werden.[28]

Es ist sicherheitsrelevant, daß die DSA-Parameter dem Verifizierer authentisch bereitgestellt werden.[29] Im Rahmen von SigI wird dies stets dadurch sichergestellt, daß die Parameter nur über Zertifikate ausgetauscht werden.

---

[28] Da die vorliegenden Spezifikation keine Änderungen der Algorithmen vorgenommen hat, wird das Prüfergebnis einer nicht SigI-konformen Komponente korrekt sein, falls die Verifikation erfolgreich beendet werden kann. Es ist nicht relevant, daß die Komponente einen Verifikationsversuch wegen einer „falschen Parameterlänge“ abbrechen kann, da Interoperabilität nur zwischen SigI-konformen Komponenten erreicht werden muß.

Der DSA-Standard erlaubt, daß die Parameter auch auf andere Art und Weise bereitgestellt werden. Möglich wäre z. B. auch, daß die Parameter für die gesamte Infrastruktur extern bekanntgemacht werden, also in keinem Zertifikat enthalten sind. SigI erlaubt dies jedoch nicht, um möglichen Bedrohungen der Authentizität der Parameter zu begegnen.[30]

3.2.3 ECDSA

Die vorliegende Spezifikation unterscheidet zwei Sigaturverfahren auf der Basis elliptischer Kurven, die auf ECDSA basieren und sich durch die Wahl des Hashalgorithmus unterscheidnen.[31]

Die Erzeugung einer digitalen Signatur erfolgt stets in folgenden Schritten:

- Bestimmung des Hashwertes der zu signierenden Daten mittels eines der in Kapitel 3.1 beschriebenen Verfahren
- Konvertierung aus einer Kodierung als Bitstring in eine Integerdarstellung
- Anwendung der Signaturfunktion mit dem privaten Schlüssel und den ECDSA-Parametern
- Konvertierung der erhaltenen Integerwerte (die Signatur besteht aus den Werten r und s) in einen Oktet-String/Bitstring
- Ausgabe der digitalen Signatur

Die Verifikation einer digitalen Signatur erfolgt stets in folgenden Schritten:

- Bestimmung des Hashwertes der zu verifizierenden Daten mittels des bei der Erzeugung der Signatur verwendeten Hashverfahrens
- Konvertierung aus einer Kodierung als Bitstring in eine Integerdarstellung
- Konvertierung der digitalen Signatur, d.h. der Werte r und s, in eine Integerdarstellung
- Anwendung der Verifikationsfunktion mit dem öffentlichen Schlüssel
- Ausgabe des Verifikationsergebnisses

3.2.3.1 Object Identifier

Für DSA werden insgesamt vier OIDs benötigt.

Folgende registrierte OIDs sind zu verwenden:

- ECDSA:

  id-publicKeyType OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) ansi-x9-62 (10045) keyType(2) 1 }[32]

---

29 Mögliche Bedrohungen, die auf einer Substitution der Parameter beruhen, sind in [Chok] beschrieben.

30 Dadurch wird auch sichergestellt, daß der Verifizierer die Parameter stets eindeutig bestimmen kann. Probleme könnten hier immer dann entstehen, wenn mehrere Arten der Bereitstellung von DSA-Parametern nebeneinander Verwendung finden würden.

31 Nach [BAZ140298] wird neben ECDSA die Verwendung weiterer Signaturverfahren auf der Basis elliptischer Kurven als geeignet angesehen. Dies gilt für ECNRA (Elliptic Curve Nyberg-Rueppel Analog Algorithms, vgl. [IEEE P1363 98], Abschnitte 7.2.5 und 7.2.6) und ECAMV (Elliptic Curve Agnew-Mullin-Vanstone analogue, vgl. [AgMuVa 93] und [ISO15946-2 99]). Im Rahmen von SigI ist die Verwendung dieser Algorithmen nicht zulässig.

- ECDSA mit SHA-1:

  ecdsa-with-sha1 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) ansi-x9-62 (10045) signatures(4) 3 }[33]

Folgende symbolischen OIDs müssen noch registriert werden:[34]

- ecdsa-extended[35]
- ecdsa-with-ripemd160

3.2.3.2 Parameter

Der ECDSA-Algorithmus verwendet folgende allgemeinen Parameter für die verwendete elliptische Kurve (vgl. Kapitel 6.3 [ANSI X9.62 99]):

- version: Integerwert für die Versionsnummer der Parameter. Für die vorliegende Spezifikation ist stets der Wert „1“ zu verwenden
- fieldID: identifiziert den endlichen Körper, über den die elliptische Kurve definiert ist
- curve: spezifiziert die Koeffizienten a und b der Kurve
- base: spezifiziert den Basispunkt G der Kurve
- order: spezifiziert die Ordnung n des Basispunktes G
- cofactor: ein Integer h

Diese Parameter können öffentlich bekannt sein. Die gleichen Parameter können mehreren Schlüsselpaaren zugeordnet sein.

Die Parameter müssen dem Verifizierer bei der Prüfung digitaler Signaturen bekannt sein. Sie müssen im Teilfeld „parameters“ des Feldes „AlgorithmIdentifier“ der Datenstruktur SubjectPublicKeyInfo des Signaturschlüssel-Zertifikats[36] des Signierers enthalten sein, sofern die Parameter nicht „vererbt“ werden.[37]

---

32 Vgl. Kapitel 6.4 [ANSI X9.62 99].

33 Vgl. Kapitel 6.5 [ANSI X9.62 99].

34 Ein TeleTrusT-OID ist für ein auf elliptischen Kurven basierendes Verfahren mit dem Hashalgorithmus RIPEMD-160 vorgesehen. Allerdings ist das Verfahren nicht exakt spezifiziert. Die folgenden OIDs kennzeichnen nur allgemein die Verwendung elliptischer Kurven:

```
ecSignWithsha1       { 1 3 36 3 3 2 1 }
ecSignWithripemd160  { 1 3 36 3 3 2 2 }
```

35 Der ECDSA-Standard erlaubt nur, daß ECDSA mit SHA-1 verwendet wird. Für die Verwendung von ECDSA mit RIPEMD-160 muß der Anwendungsbereich von ECDSA erweitert werden. Eine Begründung für die Notwendigkeit eines eigenen OIDs wird in Kapitel 3.3 gegeben.

36 Vgl. Kapitel 2.3.7 [BSI-ZERT 99].

37 Die Parameter für den öffentlichen Schlüssel werden im Standard als folgende Auswahl zwischen drei Alternativen definiert:

```
Parameters ::=          CHOICE {
        ecParamters     ECParameters,
        namedCurve      CURVES.&id({CurveNames}),
        implicitlyCA    NULL
}
```

Alle Alternativen müssen mittelfristig unterstützt werden. Hersteller müssen darauf vorbereit sein, daß OIDs für bestimmte Kurven vorgegeben werden. Sie müssen deshalb mittelfristig auch die zweite Alternative unterstützen.

Zertifizierungsstellen können die Parameter ihres eigenen Signaturschlüssel-Zertifikats optional vererben. Sofern sie Zertifikate für öffentliche Schlüssel ausstellen, deren Parameter mit ihren eigenen Parametern übereinstimmen, müssen in diesen Zertifikaten die Parameter nicht angegeben werden (vgl. [MISPC1 97]). Der Verifizierer muß die Parameter in diesem Fall aus dem übergeordneten Zertifikat der Zertifizierungsstelle entnehmen.

Auch die Wurzel-Zertifizierungsstelle kann ihre Parameter an Zertifizierungsstellen vererben, so daß in diesem Fall auch in den Zertifikaten der Zertifizierungsstellen keine Parameter enthalten sind. Eine Vererbung ist somit auch über mehrere Stufen möglich. Die Vererbung muß von allen Komponenten unterstützt werden.

Achtung: Die Forderung nach Unterstützung der Vererbung bedeutet eine Ergänzung der Vorgaben des ECDSA-Standards. Der Standard sieht vor, daß Parameter auf externem Wege bereitgestellt werden, regelt aber nicht, wie dies geschehen soll.

SigI-konforme Komponenten **müssen** in das Feld „parameters“ stets den ASN.1-Wert NULL eintragen, wenn keine Parameter angegeben werden müssen.[38]

3.2.3.3 Referenz

ECDSA ist in [ANSI X9.62 99] spezifiziert.[39]

3.2.3.4 Signaturfunktion

Der Algorithmus ist in Kapitel 5.3 [ANSI X9.62 99] beschrieben. Er besteht aus der Berechnung der elliptischen Kurve und der anschließenden Berechnung der Signatur, d.h. der Integerwerte r und s.

3.2.3.5 Verifikationsfunktion

Der Algorithmus ist in Kapitel 5.4 [ANSI X9.62 99] beschrieben. Er besteht aus einer modularen Berechnung, einer Berechnung der elliptischen Kurve und der Prüfung der Signatur.

3.2.3.6 Konvertierungen

Die Konvertierungen zwischen einem Integer und einen Oktet-String/Bitstring erfolgen entsprechend Kapitel 3.2.1.6.

Die Signatur- und Verifikationsfunktionen verwenden weitere Konvertierungen, die in Kapitel 4.3 [ANSI X9.62 99] definiert sind.

3.2.3.7 Sicherheitsanforderungen

Die Sicherheit des Verfahrens muß durch verschiedene, in [BAZ140298] genannte Bedingungen an die Parameter gewährleistet werden.

---

38 Dies entspricht den Vorgaben in Kapitel 6.4 [ANSI X9.62 99] (inplicitlyCA) und Kapitel 6.5 [ANSI X9.62 99].

39 Eine weitere, häufig verwendete Referenz ist [IEEE P1363 98]. Sie liegt bisher nur in einer vorläufigen Fassung vor. Sie unterscheidet sich im wesentlichen in der Notation von der genannten Referenz.

Für die Ordnung des Basispunktes G muß gelten: $n > 2^{160}$.[40] Als obere Grenze für die Ordnung wird im Rahmen dieser Spezifikation aus Interoperabilitätsgründen $n <= 2^{320}$ vorgegeben.

Es ist auch für die ECDSA-Parameter sicherheitsrelevant, daß sie dem Verifizierer authentisch bereitgestellt werden. Im Rahmen von SigI wird dies stets dadurch sichergestellt, daß die Parameter nur über Zertifikate ausgetauscht werden.

Der ECDSA-Standard erlaubt, daß die Parameter auch auf andere Art und Weise bereitgestellt werden. Möglich wäre z. B. auch, daß die Parameter für die gesamte Infrastruktur extern bekanntgemacht werden, also in keinem Zertifikat enthalten sind. SigI erlaubt dies jedoch nicht, um möglichen Bedrohungen der Authentizität der Parameter zu begegnen.

3.3 Verwendung der Algorithmen-Identifizierer

OIDs sind ASN.1-Objekte vom Typ OBJECT IDENTIFIER. Sie werden im Rahmen von SigI an vielen Stellen zur Identifizierung von Algorithmen verwendet. Die Identifierung durch einen OID erfolgt stets durch Angabe im Feld „algorithm“ der Datenstrukur „AlgorithmIdentifier“[41].

OIDs werden in SigI verwendet zur:

- Identifizierung eines Hashalgorithmus
- Identifizierung eines Signaturverfahrens exklusive des Hashalgorithmus
- Identifizierung eines Signaturverfahrens insgesamt, d.h. inklusive Hashalgorithmus

Die Algorithmen-Identifizierer werden in SigI nicht einheitlich verwendet. Dies beruht darauf, daß die Standards, auf denen SigI basiert, die Verwendung unterschiedlich regeln. Dabei ist zwischen Anwendungen zu unterscheiden, die auf der CMS-Syntax [CMS 98] basieren und allen sonstigen Anwendungen.[42]

Im folgenden wird die Verwendung von Algorithmus-Identifizierern in Zertifikaten und digital signierten Nachrichten dargestellt. Auf diese Typen lassen sich alle anderen Anwendungen zurückführen.

3.3.1 Verwendung in Zertifikaten

Identifizierer werden in Zertifikaten verwendet, um das Signaturverfahren insgesamt zu identifizieren, das bei der Signatur des Zertifikats selbst verwendet wurde. Diese Verwendungsart ist nicht auf Zertifikate beschränkt.

Zusätzlich werden Identifizierer in Zertifikaten verwendet, um für den öffentlichen Schlüssel, dessen Zuordnung zu einer natürlichen Person im Zertifikat bescheinigt wird, den Algorithmus anzugeben, mit dem er verwendet werden muß.

---

40 Vgl. Kapitel 5.1 [ANSI X9.62 99].

41 Der Typ „AlgorithmIdentifier“ hat folgende ASN.1-Struktur:

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

42 Auf der CMS-Syntax basieren digital signierte Nachrichten und Zeitstempel.

3.3.1.1 Identifizierung des Signaturverfahrens

Zur Identifizierung des Signaturverfahrens werden in Zertifikaten OIDs in den Feldern „signatureAlgorithm“ und „signature“ eingetragen (vgl. Kapitel 2 [BSI-ZERT 99]). Die Einträge in diesen Feldern identifizieren stets das gesamte Signaturverfahren inklusive Hashalgorithmus.

Für SigI ergibt sich keine Abweichung von den Standards. Es müssen lediglich noch weitere OIDs für die Kombinationen mit RIPEMD-160 und dem ISO9796-2rnd-Padding registriert werden.

3.3.1.2 Identifizierung des Algorithmus für den Schlüssel

In einem standardkonformen X.509-Zertifikat ist der öffentliche Schlüssel stets zusammen mit einem OID enthalten, der das Signaturverfahren exklusive des Hashalgorithmus identifiziert, für das der Schlüssel Verwendung finden soll.[43]

Dies bietet den Vorteil, daß nicht bereits durch das Zertifikat vollständig festgelegt wird, für welches Signaturverfahren der öffentliche Schlüssel zu verwenden ist. Falls bei der Generierung von Signaturen z.B. sowohl SHA-1 als auch RIPEMD-160 verwendet werden soll, wird nur ein Zertifikat benötigt.

In SigI muß hiervon jedoch teilweise abgewichen werden. Dies ist eine Folge der bereits beschriebenen Erweiterungen der Standards (Zulassung von RIPEMD-160 für DSA und ECDSA) und der Regelungen in der Chipkartenspezifikation des DIN [DINChip 98]. Die Ursache ist stets, daß der verwendete Hashalgorithmus sicher an die Signatur gebunden werden und deshalb bereits über Informationen im Zertifikat eine Bindung zwischen Signaturfunktion und Hashalgorithmus erfolgen muß.

Die Bindung zwischen Signaturfunktion und Hashalgorithmus ist stets erforderlich, um möglichen Bedrohungen zu begegnen. Ohne eine solche Bindung könnte ein Angreifer[44] den Identifikator für den Hashalgorithmus gegen einen Identifikator für einen schwachen Hashalgorithmus austauschen. Der Angreifer könnte dann eine Nachricht konstruieren, die unter Verwendung des schwachen Hashalgorithmus den gleichen Hashwert liefert und fälschlicherweise behaupten, die digitale Signatur sei für diese Nachricht unter Verwendung des schwachen Hashalgorithmus generiert worden.[45]

---

43 In X.509-Zertifikaten ist ein öffentlicher Schlüssel stets in folgender Datenstruktur enthalten (vgl. Kapitel 8 [ITU-T X.509 97]):

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

44 Als Angreifer kommt jeder in Betracht, der daraus einen Vorteil ziehen kann, daß die Echtheit der digitalen Signatur in Zweifel gezogen wird. Angreifer kann somit der Signierer sein, der die Echtheit einer von ihm erstellten Signatur bestreitet oder z.B. der Empfänger einer Willenserklärung, der nachträglich behauptet, ein Vertrag sei nicht wirksam zustande gekommen.

45 Im folgenden Beispiel sei H ein starker und h ein schwacher Hashalgorithmus, wobei h ursprünglich ebenfalls als geeigneter starker Algorithmus angesehen worden ist, dessen Schwäche sich erst nachträglich herausgestellt hat. Der Signierer bildet nun zwei verschiedene Datensätze als Input für die Hashalgorithmen so, daß gilt: H(data1)=h(data2). Da h schwach ist, ist dies stets möglich.

Falls in die anschließende Signaturbildung der tatsächlich verwendete Hashalgorithmus nicht eingeht, kann der Empfänger die Nachricht nur anhand des ungesichert übertragenen Identifikators verifizieren. Dies kann der Signierer ausnutzen, um dem Empfänger durch Angabe des Identifikators H und des Datensatzes data1 zunächst davon zu überzeugen, er habe data1 signiert,

Ob der Angreifer mit dieser Behauptung Erfolg hat, hängt letztlich von den Umständen des Einzelfalles ab. Das Ziel des Signaturgesetzes, jede Art von Fälschung digitaler Signaturen und Verfälschungen signierter Daten zuverlässig feststellen zu können, würde jedoch nicht erreicht.

Für eine sichere Bindung des Hashalgorithmus gibt es folgende Alternativen:[46]

- Am sichersten ist es, zu fordern, daß eine Verifikationskomponente stets einen bestimmten Hashalgorithmus verwenden muß. Diese Alternative wird bei den Standards für DSA und ECDSA verwendet.[47] Sie kommt jedoch nach der vorgenommenen Erweiterung der Standards für SigI nicht mehr in Betracht.
- Die zweitsicherste Methode ist es, den Identifikator für den Hashalgorithmus in die Signatur einzubeziehen. Beim RSA-Verfahren mit der Padding-Variante PKCS erfolgt die Einbeziehung eines Algorithmen-Identifizierers in den Padding-String.[48] Beim ISO9796-2rnd-Padding ist eine solche Einbeziehung nicht vorgesehen.
- Eine weitere Methode ist, im Zertifikat die Verwendung mehrerer Hashfunktionen zuzulassen, jedoch für die gesamte PKI nur die Verwendung einer dieser Hashfunktionen zuzulassen.[49] Diese Alternative ist ausgeschlossen, da keine Beschränkung auf einen Hashalgorithmus erfolgen soll.
- Eine Beschränkung auf einen Hashalgorithmus muß nicht für die gesamte PKI vorgenommen werden. Die Beschränkung kann auch durch eine Vereinbarung zwischen Signierer und Verifizierer erfolgen. Diese Alternative ist ebenfalls ausgeschlossen, da sie die zuverlässige Verifikation digitaler Signaturen von der zuverlässigen Kenntnis des vereinbarten Hashalgorithmus abhängig macht. Darüber hinaus wären Sicherheitsprobleme zu befürchten, da der Verifizierer vor jeder Prüfung die für die Prüfung zu verwendende Hashfunktion einstellen müßte und vorhersehbar ist, daß dem Verifizierer dabei Fehler unterlaufen.

Als Ergebnis kann festgehalten werden, daß eine standardkonforme Verwendung der OIDs zur Identifizierung des Signaturverfahrens exklusive des Hashalgorithmus in Zertifikaten nur bei RSA mit PKCS-Padding in Betracht kommt.

---

dies später jedoch abzustreiten, wenn er dadurch einen Vorteil erlangen kann. (Bei Angabe einer schwachen Hashfunktion würde seitens der Empfängerkomponente eine Meldung gemäß [BSI-GM 99] generiert, die die fehlende Eignung der Algorithmen betrifft. Die Verwendung einer schwachen Hashfunktion würde der Empfänger daher erkennen und die Signatur zurückweisen.)

Der Signierer könnte z.B. später behaupten, er habe h als Indikator für den Hashalgorithmus angegeben und data2 signiert. Es habe nicht gewußte, daß h bereits zum Signaturzeitpunkt schwach gewesen sei, obwohl er sich regelmäßig davon überzeugt habe, daß die von ihm verwendeten Hashalgorithmen als geeignet angesehen werden. Die Tatsache, daß in der vom Empfänger präsentierten digital signierten Nachricht als Identifikator für den Hashalgorithmus H ausgewiesen sei und sich diese auf data1 bezieht könne er sich nur so erklären, daß die Nachricht auf dem Transportwege unerkannt manipuliert worden sei oder daß der Empfänger selbst diese Manipulation vorgenommen habe.

46 Die Beschreibung dieser Alternativen erfolgt entsprechend Kapitel 7 [ISO 14888-3 98].

47 Zu beachten ist, daß es auch bei DSA und ECDSA unterschiedliche OIDs zur Idenfizierung des Signaturverfahren inklusive und exklusive Hashalgorithmus gibt.

48 Bei PKCS wird aus dem OID und dem Hashwert die Datenstrukur „DigestInfo“ gebildet und anschließende mit einem speziellen Padding-String konkateniert (vgl. Kapitel 9.2.1 [PKCS1 98]).

49 Die Wurzel-Zertifizierungsstelle müßte in diesem Fall die Information über den zu verwendenden Hashalgorithmus allgemein bekanntgeben.

In allen anderen Fällen (RSA mit ISO9796-2rnd-Padding, DSA und ECDSA) muß der Verifikationskomponente über Informationen im Zertifikat mitgeteilt werden, welchen Hashalgorithmus sie zu verwenden hat. Um dem Standard möglichst nahe zu kommen wird diese Information im Rahmen von SigI stets über einen speziellen OID übermittelt, der in das Zertifikatsfeld subjectPublicKeyInfo eingetragen wird. Durch Auswertung des OIDs kann die Verifikationskomponente erkennen, welchen Hashalgorithmus sie verwenden muß.

Achtung: Die Spezifikation enthält in Abweichung von den Standards OIDs zur Identifizierung von Signaturverfahren exklusive Hashverfahren, bei denen die Verifikationskomponente sicherstellen muß, daß das Signaturverfahren mit dem im Signaturschlüssel-Zertifikat bestimmten Hashalgorithmus verwendet wurde.

Für SigI sind die folgenden OIDs zu verwenden:

- Für RSA mit ISO9796-2rnd-Padding ist einer der zwei OIDs „rsa-padding-iso9796-2rnd-indicate-sha1 oder rsa-padding-iso9796-2rnd-indicate-ripemd160“ zu verwenden, die der Verifikationskomponente indizieren, welchen Hashalgorithmus sie verwenden muß.
- Für DSA ist der bereits registrierte OID zu verwenden, um der Verifikationskomponente zu indizieren, daß sie den Hashalgorithmus SHA-1 verwenden muß. Dies entspricht dem Standard bei DSA. Durch den OID „dsa-extended“ wird der Verifikationskomponente indiziert, daß sie statt dessen RIPEMD-160 verwenden muß.
- Für ECDSA gilt das gleiche wie für DSA.

3.3.2 Verwendung in digital signierten Nachrichten

Algorithmen-Identifizierer werden in digital signierten Nachrichten verwendet, um den Signaturalgorithmus zu identifizierten, mit dem die Nachricht signiert wurde (vgl. Kapitel 12.2 [CMS 98]). In CMS werden für RSA und DSA typmäßig unterschiedliche OIDs verwendet.[50] Für RSA wird ein OID verwendet, der das Signaturverfahren exklusive Hashalgorithmus identifiziert während für DSA das Signaturverfahren inklusive Hashalgorithmus identifiert wird.

Der in [CMS 98] angegebene OID für RSA betrifft RSA mit PKCS-Padding und der OID für DSA betrifft das DSA-Verfahren mit Verwendung von SHA-1. Um unnötige Abweichungen von Standards zu vermeiden müssen diese OIDs auch in SigI verwendet werden.

Für SigI sind für die weiteren zugelassenen Verfahren folgende OIDs zu verwenden:

- Für RSA mit ISO9796-2rnd-Padding ist einer der zwei OIDs „rsa-padding-iso9796-2rnd-indicate-sha1 oder rsa-padding-iso9796-2rnd-indicate-ripemd160“ zu verwenden, die der Verifikationskomponente indizieren, welchen Hashalgorithmus sie verwenden muß (s. Kapitel 3.3.1).
- Für DSA mit RIPEMD-160 ist der OID „dsa-with-ripemd160“ zu verwenden
- Für ECDSA mit SHA-1 ist der OID „ecdsa-with-sha1“ zu verwenden
- Für ECDSA mit RIPEMD-160 ist der OID „ecdsa-with-ripemd160“ zu verwenden

3.4 Zusammenstellung der Object Identifier

Es gibt keine Organisation, die alle im Rahmen der Spezifikation benötigten OIDs registriert hat. Deshalb müssen OIDs verschiedener Organisationen verwendet werden. Falls mehrere Organisationen einen OID vergeben haben, wird der OID verwendet, der am gebräuchlichsten

---

50 ECDSA wird in [CMS 98] nicht behandelt.

ist. Falls bisher noch kein OID vergeben wurde, wird ein vorläufiger symbolischer OID angegeben.

In den folgenden Tabellen werden die OIDs für die im Rahmen der Spezifikation zugelassenen Hashalgorithmen sowie die Signaturverfahren inklusive und exklusive Hashalgorithmus angegeben. Die erste Spalte enthält jeweils des Algorithmus, die zweite den OID und die letzten Spalte die den OID verwaltende Organisation, sofern ein OID bereits vergeben wurde.

3.4.1 Hashalgorithmen

SHA-1	{ 1 3 14 3 2 26 }	OIW
RIPEMD-160	{ 1 3 36 3 2 1 }	TTT

Tabelle 1: Hashalgorithmen

3.4.2 Signaturverfahren exklusive Hashalgorithmus

RSA mit PKCS-Padding	{ 1 2 840 113549 1 1 1 }	rsadsi (RSA Laboratories)
RSA mit ISO9796-2rnd-Padding und Indizierung von SHA-1	rsa-padding-iso9796-2rnd-indicate-sha1	
RSA mit ISO9796-2rnd-Padding und Indizierung von RIPEMD-160	rsa-padding-iso9796-2rnd-indicate-ripemd160	
DSA mit Indizierung von SHA-1	{ 1 2 840 10040 4 1 }	ANSI X9.57
DSA mit Indizierung von RIPEMD-160	dsa-extended	
ECDSA mit Indizierung von SHA-1	{ 1 2 840 10045 2 1 }	ANSI X9.62
ECDSA mit Indizierung von RIPEMD-160	ecdsa-extended	

Tabelle 2: Signaturverfahren exklusive Hashalgorithmus

3.4.3 Signaturverfahren inklusive Hashalgorithmus

RSA mit PKCS-Padding und SHA-1	{ 1 2 840 113549 1 1 5 }	rsadsi (RSA Laboratories)
RSA mit PKCS-Padding und RIPEMD-160	{ 1 3 36 3 3 1 2 }	TTT
RSA mit ISO9796-2rnd-Padding und SHA-1	rsa-padding-iso9796-2rnd-with-sha1	
RSA mit ISO9796-2rnd-Padding und RIPEMD-160	rsa-padding- iso9796-2rnd-with-ripemd160	
DSA mit SHA-1	{ 1 2 840 10040 4 3 }	ANSI X9.57

DSA mit RIPEMD-160	dsa-with-ripemd	
ECDSA mit SHA-1	{ 1 2 840 10045 4 3 }	ANSI X9.62
ECDSA mit RIPEMD-160	ecdsa-with-ripemd	

Tabelle 3: Signaturverfahren inklusive Hashalgorithmus

4 Signaturumfang

Im Rahmen von SigI werden verschiedene Arten digital signierter Daten unterschieden. Digitale Signaturen werden sowohl zum gesicherten Austausch von Informationen zwischen zwei Teilnehmern der Sicherheitsinfrastruktur als auch zwischen Zertifizierungsstellen und Teilnehmern verwendet. Der gesicherte Austausch von Information zwischen Teilnehmern wird im folgenden dargestellt.

Zum Austausch von Informationen zwischen Teilnehmern wird eine **Nachricht** gebildet, die ein definiertes Format hat. Die Nachricht besteht neben den digitalen Daten aus weiteren Daten, die für den Transport der Nachricht und die Prüfung der digitalen Daten benötigt werden. Zur Unterscheidung werden die digitalen Daten als eigentlicher Inhalt der Nachricht im folgenden als **Nutzdaten** bezeichnet.

Das Format für die Nachrichten insgesamt wird im folgenden als **Austauschformat** bezeichnet. Der Begriff **Signaturaustauschformat** wird verwendet, um das Format für die Integration der Nutzdaten, der Signatur, der für die Prüfung der Signatur erforderlichen Daten und sonstiger Daten, die sich auf die Nutzdaten beziehen, zu identifizieren. Bei der Generierung einer signierten Nachricht wird stets zunächst ein durch eine digitale Signatur gesichertes Datenobjekt erzeugt, das dem Signaturaustauschformat entspricht. Dieses Datenobjekt soll im folgenden als **Signaturaustauschobjekt** bezeichnet werden.

Um die Nachricht austauschen zu können, muß das Signaturaustauschobjekt anschließend noch in ein Transportformat eingebettet werden. Das Austauschformat definiert somit ein Transportformat mit integriertem Signaturaustauschformat. Datenobjekte vom Typ Austauschformat werden im folgenden als **Austauschobjekt** bezeichnet.

Nachrichten bestehen aus einer Vielzahl einzelner Daten. Nicht alle dieser Daten müssen authentisch und unverfälscht übertragen und deshalb in die Signatur eingeschlossen werden. Neben den Nutzdaten selbst sind jedoch stets weitere Daten des Signaturaustauschobjekts in die Signatur einzubeziehen.

Es müssen notwendigerweise alle Daten in die Signatur einbezogen werden, die benötigt werden, um die Signatur zuverlässig prüfen zu können. Dies betrifft auch Informationen über den Typ der Nutzdaten, der durch die digitale Signatur gesichert übertragen werden muß. Andernfalls könnte der Empfänger der Nachricht über den Typ der Daten getäuscht werden. Der „Typ der digitalen Daten“ ist daher stets in die Signatur einzubeziehen.

Sicherheitsanforderungen in SigG/SigV bestimmen entweder explizit oder implizit, welche Daten gesichert übertragen werden müssen. Diese Daten gehören obligatorisch zum **Signaturumfang**, sofern sie nicht mittelbar in die Signatur einbezogen sind. Die Informationen in Signaturschlüssel-Zertifikaten können z.B. mittelbar über eine eindeutige Referenz auf das Zertifikat in die Signatur einbezogen sein. In diesem Fall ist der für die Signaturbildung verwendete Signaturalgorithmus als Teil des Zertifikates mittelbar in die Signatur einbezogen.

Der Signierer kann neben den Daten, die zwingend in eine Signatur einzubeziehen sind, weitere optionale Informationen in ein Signaturaustauschobjekt integrieren und in die Signatur einbeziehen. Die betrifft z.B. den Ort der Signaturerstellung wie bei Papierdokumenten.

Signaturgesetz-konforme Applikationen müssen die Einbeziehung dieser optionalen Informationen nicht unterstützen, da entsprechende Anforderungen aus SigG/SigV nicht abgeleitet werden können.[51] Die Aufnahme optionaler Informationen in die Spezifikation erfolgt, um den Herstellern von Applikationen die Unterstützung weiterer sinnvoller Funktionen zu erleichtern.

Herstellern erlaubt die Spezifikation somit, zusätzliche Informationen auf eine definierte Art und Weise in das Signaturaustauschobjekte zu integrieren. Absender, die von den Optionen Gebrauch machen wollen, müssen nur sicher sein, daß auch die Applikation auf Empfängerseite die Option unterstützt. Das „Wie“ der Unterstützung wird dann durch die vorliegende Spezifikation bestimmt.

---

51 Gefordert wird allerdings, daß sie optionalen Daten ignorieren, falls sie sie nicht auswerten können oder wollen (s. Kapitel 4.2).

Die nachfolgende Tabelle gibt einen Überblick über die relevanten Daten, wobei sie danach gegliedert ist, ob die Daten obligatorische oder optionale Teile des Signaturaustauschformats sind und danach, ob die beigefügten Daten sich auf die Nutzdaten oder auf die Signatur beziehen:

	Obligatorische Daten	Optionale Daten
Nutzdaten		
Inhalt und Typ der Nutzdaten		
Quittungsanforderung		
Dateibezeichner		
Speicherdatum und -zeit		
Dateigröße		
Signatur		
Signaturschlüssel-Zertifikat bzw. Referenz darauf	[52]	
Attribut-Zertifikate bzw. Referenz darauf	[53]	
Signaturverfahren		
Hashverfahren		
Paddingverfahren		
Signaturdatum und -zeit		
Ort		
Signaturnummer		
Automatisch erstelle Signatur		
Mitzeichnung		

Tabelle 4: Signaturumfang

4.1 Obligatorische Daten

Die folgenden Daten sind obligatorische Teile des Signaturaustauschformats und müssen stets in die Signatur einbezogen werden:

4.1.1 Inhalt und Typ der Nutzdaten

Die Nutzdaten, der eigentliche Inhalt einer digital signierten Nachricht, sind in der Regel Teil jedes Signaturaustauschobjekts. Nur Ausnahmsweise sind die Nutzdaten nicht Teil des Signaturaustauschobjektes, wenn eine sog. externe Signatur gebildet wird. Bei externen

---

52 Enthält das Signaturschlüssel-Zertifikat Beschränkungen nach § 7 Abs. 1 Nr. 7 oder § 7 Abs. 2 SigG, so ist das Zertifikat selbst obligatorisch in den Signaturumfang einzuschließen. Dies gilt auch, wenn das Zertifikat nicht abrufbar ist. Andernfalls muß nur eine Referenz auf das Zertifikat mitsigniert werden.

53 Enthält ein Attribut-Zertifikat Beschränkungen nach § 7 Abs. 1 Nr. 7 oder § 7 Abs. 2 SigG, so ist das Zertifikate selbst obligatorisch in den Signaturumfang einzuschließen. Dies gilt auch, wenn das Zertifikat nicht abrufbar ist. Andernfalls muß nur eine Referenz auf das Zertifikat mitsigniert werden.

Signaturen sind die Nutzdaten und die Signatur getrennt in verschiedenen Nachrichten oder Nachrichtenteilen enthalten.[54]

Aus Gesetz und Verordnung läßt sich ableiten, daß daneben auch der Typ der Nutzdaten in die Signatur einbezogen werden muß. Die digitale Signatur der Nutzdaten muß nach § 2 Abs. 2 und § 14 Abs.2 Satz 1 SigG so beschaffen sein, daß ein Verifizierer die Unverfälschtheit der Nutzdaten erkennen kann. Eine erfolgreiche Fälschung setzt keinesfalls voraus, daß die Signatur oder die Nutzdaten manipuliert werden. Auch die semantische Unverfälschtheit der Daten muß durch die digitale Signatur gewährleistet werden (vgl. [Fox 98]).

Die semantische Unverfälschtheit der Daten ist gewährleistet, wenn die Daten durch die Darstellungskomponenten des Signierers und der Verifizierers identisch angezeigt werden. Die Darstellungskomponente des Verifizieres muß dazu zumindest den Datentyp kennen. Der Datentyp der Nutzdaten muß daher in die digitale Signatur einbezogen werden.[55]

Die technische Realisierung besteht darin, daß die Nutzdaten mit einem Identifikator (Object Identifier oder kurz OID) versehen werden, der den Typ der Daten angibt. Eine vorläufige Liste von OIDs für Datentypen ist in Anlage A enthalten.

Ein spezieller OID (id-data) ist zu verwenden, wenn eine Typinformation den Nutzdaten nicht zugewiesen werden kann oder soll. Bei Verwendung dieses OIDs ergibt die Typinformation nur, daß es sich um beliebige binäre Daten handelt. Der Verifizierer muß dann gegebenenfalls die erforderliche Information auf andere, nicht weiter spezifizierte Weise beim Signierer einholen.

Durch einen OID können neben der Typangabe weitere Informationen für die Darstellungskomponente gesichert übertragen werden. Da es eine unüberschaubare Vielzahl von Darstellungskomponenten mit einer Vielzahl von Parametern gibt, die für die Darstellung von Bedeutung sind[56], ist es nicht sinnvoll, in der vorliegenden Spezifikation entsprechende OIDs vorzugeben.

Nur anhand der konkreten Anwendungen läßt sich entscheiden, welche Informationen ein OID liefern soll. Falls ein OID in Anhang A die erforderlichen Informationen enthält, soll er verwendet werden. Sofern dies nicht der Fall ist, können jederzeit neue OIDs registriert werden, die den Anforderungen der konkreten Anwendung gerecht werden.

4.1.2 Signaturschlüssel-Zertifikat

Der Verifizierer einer digitalen Signatur benötigt das Signaturschlüssel-Zertifikat des Signierers, um diesem neben weiteren Informationen den authentischen öffentlichen Schlüssel zur Verifikation der digitalen Signatur zu entnehmen. Das Zertifikat kann er entweder vom Signierer oder vom Verzeichnisdienst der ZS erhalten. Im letztgenannten Fall muß das Zertifikat eindeutig so identifiziert werden, daß eine Verzeichnisdienstabfrage möglich ist, also durch Angabe des technischen Namens der ZS und der Seriennummer des Zertifikats.

---

54 Dies hat den Vorteil, daß beide Nachrichten unterschiedlich kodiert werden können. Die Nachricht mit den Nutzdaten kann z.B. so kodiert werden, daß die Daten auch mit Applikationen darstellbar sind, die die Kodierungen des Signaturaustauschformates nicht erkennen können. Damit kann die Nachricht mit diesen Applikationen zwar nicht verifiziert, jedoch angezeigt werden.

55 Es ist zu beachten, daß dies zwar eine notwendige, aber nicht in allen Fällen auch eine hinreichende Bedingung für die semantische Unverfälschtheit ist. Der Typ der Daten ist nur ein Parameter von vielen, die die Darstellung der Daten beeinflussen können.

56 Beispiele in [Fox 98] sind: Spezielle Anzeigeoptionen (wie die Darstellung von verborgenem Text oder eine nahezu unsichtbare Farbwahl einzelner Zeichen) oder das verwendete Betriebsystem.

Für den Signierer besteht grundsätzlich Wahlfreiheit, ob er sein Zertifikat der Signatur beifügt oder ob er das Zertifikat nur identifiziert. Falls das Zertifikat jedoch Beschränkungen nach § 7 Abs. 1 Nr. 7 oder § 7 Abs. 2 SigG enthält, muß es gemäß § 4 Abs. Nr. 4 SigV stets den Daten beigefügt und in die Signatur eingeschlossen werden.[57]

Auch in einem weiteren Fall muß im Rahmen dieser Spezifikation stets das Signaturschlüssel-Zertifikat und nicht bloß eine Referenz auf das Zertifikat Teil der Nachricht sein und in die Signatur eingeschlossen werden. Gemäß § 5 Abs. 1 SigG darf eine ZS ein Signaturschlüssel-Zertifikat nur dann abrufbar halten, wenn der Signaturschlüssel-Inhaber zugestimmt hat. Der Inhaber kann sich somit durch Nichterteilung dieser Zustimmung vorbehalten, das Zertifikat selbst zu verteilen. In diesem Fall kann das Signaturschlüssel-Zertifikat nicht von der ZS bezogen werden, d.h. es ist nicht abrufbar.[58]

Damit eine Verifikation digitaler Signaturen im Falle nicht abrufbarer Signaturschlüssel-Zertifikate möglich ist, muß das Zertifikat bei nicht abrufbaren Zertifikaten in der Regel Teil des Signaturaustauschobjektes sein.[59] Nur dieser Regelfall ist Gegenstand der vorliegenden Spezifikation.

Im Rahmen der Spezifikation wird vom Signierer gefordert, in den genannten Fällen das Zertifikat beizufügen. Falls er dies unterläßt, trägt er das Risiko dafür, das seine digitale Signatur als „nicht prüfbar“ und damit als nicht Signaturgesetz-konform zurückgewiesen wird.[60]

4.1.3 Referenz auf Signaturschlüssel-Zertifikat

Falls kein Signaturschlüssel-Zertifikat in die Signatur eingeschlossen wird muß stets eine Referenz auf dieses Zertifikat Teil der Signatur sein. Eine Referenz auf ein Signaturschlüssel-Zertifikat besteht aus einer Seriennummer und dem technischen Namen der ZS.[61]

---

57 In die Signatur muß stets das Zertifikat selbst eingeschlossen werden und nicht bloß eine Referenz auf das Zertifikat. Es bleibt dahingestellt, ob eine andere Interpretation mit dem Signaturgesetz vereinbar wäre. Die vorliegende Interoperabilitätsspezifikation beschränkt sich jedenfalls auf diese Variante.

58 In einer Anfrage an den Verzeichnisdienstes muß die SigI-Erweiterung „retrieveIfAllowed“ gesetzt sein, damit der Verzeichzeichnisdienst das Zertifikat statt bloß die Referenz auf das Zertifikat liefert (vgl. Kapitel 2.2.1 [BSI-DIR 99] , „Anfragen an den Verzeichnisdienst“). Der Verzeichnisdienst liefert nur die Referenz des Zertifikates, falls das Zertifikat nicht abrufbar ist (vgl. Kapitel 2.2.2 [BSI-DIR 99] , „Antworten des Verzeichnisdienstes“).

59 Ausnahmsweise könnte darauf verzichtet werden, falls der Verifizierer bereits im Besitz des Zertifikates ist, weil das Zertifikat z.B. Teil einer früheren Nachricht war und vom Verifizierer gespeichert wurde.

60 Signaturen, die nur unter der Bedingung prüfbar sind, daß ein Verifizierer bereits im Besitz des Signaturschlüssel-Zertifikats des Signierers ist, werden als nicht Signaturgesetz-konform angesehen. Eine digitale Signatur nach § 2 Abs. 1 SigG muß stets die Unverfälschtheit der Daten erkennen lassen. Eine Voraussetzung dafür ist ihre Prüfbarkeit, die nicht nur für den Adressaten einer digital signierten Nachricht selbst, sondern aus Gründen der für digitale Signaturen geforderten Nichtabstreitbarkeit auch für einen beliebigen Dritten ohne weiteres gegeben sein muß.

Da die vorliegende Spezifikation nur Signaturgesetz-konforme Signaturen behandelt, muß der Signierer nicht abrufbare Zertifikate stets der Signatur beifügen.

61 Zertifikate werden entsprechend Kapitel 2.3.2 [BSI-ZERT 99] durch die Kombination aus der Seriennummer (serialNumber) und dem Namen der Zertifizierungsstelle (issuer) global eindeutig identifiziert.

Durch die Einbeziehung der Referenz wird eindeutig und unfälschbar bestimmt, welches Zertifikat der Verifizierer bei der Gültigkeitsprüfung der digitalen Signatur zugrundelegen muß.[62]

4.1.4 Attribut-Zertifikate

Ein Attribut-Zertifikat ist gemäß § 2 Abs. 3 SigG ein Zertifikat, das sich auf ein Signaturschlüssel-Zertifikat bezieht. Für Attribut-Zertifikate gilt das gleiche wie für Signaturschlüssel-Zertifikate. Falls das Attribut-Zertifikat Beschränkungen enthält oder nicht abrufbar ist, muß es Teil des Signaturaustauschobjektes sein und in die digitale Signatur einbezogen werden.

4.1.5 Referenz auf Attribut-Zertifikate

Für Attribut-Zertifikate gilt das gleiche wie für das Signaturschlüssel-Zertifikat. Eine Referenz auf ein Signaturschlüssel-Zertifikat besteht mindestens aus einer Seriennummer und dem technischen Namen der ZS. Die Referenz muß Teil des Signaturaustauschobjektes sein und in die Signatur einbezogen werden.

4.1.6 Signaturverfahren

Das verwendete Signaturverfahren muß ebenfalls unfälschbar mit der digitalen Signatur verbunden sein. Das Signaturverfahren ist stets indirekt in die Signatur einbezogen.[63] Die Unfälschbarkeit des Signaturschlüssel-Zertifikats wird durch die Signatur der ausstellenden ZS garantiert.

4.1.7 Hashverfahren

Auch das verwendete Hashverfahren muß unfälschbar mit der digitalen Signatur verbunden sein. In SigI wird das Hashverfahren stets direkt oder indirekt in die Signatur einbezogen. Eine direkte Einbeziehung erfolgt bei RSA mit der Padding-Variante PKCS. In allen anderen Fällen erfolgt eine indirekte Einbeziehung indem der Verifikationskomponente indiziert wird, welchen Hashalgorithmus sie verwenden muß.[64]

4.1.8 Padding-Verfahren

Für RSA sind zwei Paddingverfahren spezifiziert. Bei der Verifikation muß bekannt sein, welches Padding-Verfahren zu verwenden ist. Das Padding-Verfahren ist stets indirekt in die Signatur einbezogen.[65]

---

62 Es ist nicht ausgeschlossen, daß ein Nutzer mehrere Signaturschlüssel-Zertifikate für den gleichen Schlüssel erhält. In diesem Fall gibt es mehrere Zertifizierungspfade. Durch eine Fälschung der Referenz auf das Zertifikat würde der Verifizierer den falschen Zertifizierungspfad prüfen, ohne dies erkennen zu können.

63 Ein Signaturschlüssel-Zertifikat verknüpft einen öffentlichen Schlüssel mit einem Signaturverfahren durch Verwendung der Datenstruktur „SubjectPublicKeyInfo“ (vgl. Kapitel 2.3.7 [BSI-ZERT 99]). Das Signaturverfahren ist somit durch das Zertifikat indirekt in die Signatur einbezogen.

64 Vgl. Kapitel 3.3.1.2.

65 Ein Signaturschlüssel-Zertifikat verknüpft einen öffentlichen Schlüssel mit einem Signaturverfahren inklusive Paddingverfahren durch Verwendung der Datenstruktur

4.2 Optionale Daten

Optionale Daten können Teil des Signaturaustauschobjektes sein. Der Signierer kann dadurch bestimmte Informationen über die Nutzdaten oder die Signatur austauschen. Es liegt in seinem freien Ermessen, von dieser Möglichkeit Gebrauch zu machen.

Aus SigG/SigV können keine entsprechenden Anforderungen für die Unterstützung der optionalen Daten abgeleitet werden. Signaturgesetz-konforme Applikationen müssen optionale Daten daher weder generieren noch auswerten können. Gefordert wird allerdings, daß sie optionalen Daten ignorieren, falls sie sie nicht auswerten können oder wollen.

Grundsätzlich sollen auch optionale Daten in die Signatur einbezogen werden. Der Signierer erhält keine Wahlfreiheit hinsichtlich der Einbeziehung in die digitale Signatur. Eine Ausnahme von diesem Grundsatz muß bei der Mitzeichnung gemacht werden. Alle anderen optionalen Daten müssen in die Signatur einbezogen werden.

4.2.1 Quittungsanforderung

Im Rechtsverkehr ist neben der Echtheit einer Willenserklärung in der Regel der Zugang der Erklärung beim Empfänger von wesentlicher Bedeutung. Der Absender einer Willenserklärung muß im Streitfall beweisen können, daß der Empfänger die Erklärung tatsächlich erhalten hat. Ein geeignetes Beweismittel hierfür ist eine Quittung des Empfängers.

Im Rechtsverkehr kann es für den Absender deshalb erforderlich sein, der signierten Willenserklärung eine Quittungsanforderung beizufügen. Digitale Quittungsanforderungen sind in [ESS 99] definiert.

4.2.2 Dateibezeichner

Ein Dateibezeichner (mit oder ohne Pfadangabe) kann für die Weiterverarbeitung beim Empfänger von Bedeutung sein. Es sind mehrere Fälle zu unterscheiden.

Wie bereits beschrieben können im Falle externer Signaturen die Signatur und eine Klartextnachricht in verschiedenen Teilnachrichten untergebracht werden. Der Absender kann vorschlagen, daß z.B. ein Attachment in einer separaten Datei gespeichert wird. Falls der Empfänger das Attachment abspeichert, sollte der aktuelle Dateibezeichner dem vorgeschlagenen Dateibezeichner entsprechen, falls möglich. Eine anschließende Kommunikation zwischen den Parteien wird dann dadurch vereinfacht, daß die Datei über den gleichen Bezeichner referenziert werden kann.

Bei einer weiteren Anwendung externer Signaturen werden die Nutzdaten nicht in einer weiteren Teilnachricht ausgetauscht, sondern in einer eigenen Datei. Die Datei kann unabhängig vom der Nachricht selbst ausgetauscht werden. Der Empfänger kann sie z.B. von einer Web-Seite laden. Durch den Dateibezeichner kann der Empfänger in diesem Fall erkennen, auf welche Datei sich die digitale Signatur bezieht.

4.2.3 Speicherdatum und -zeit

Datum und Zeit der letzten Speicherung eines Dokumentes können dem Empfänger einen Hinweis darauf geben, wann eine Datei generiert wurden.

„SubjectPublicKeyInfo“ (vgl. Kapitel 2.3.7 [BSI-ZERT 99]). Das Paddingverfahren ist somit durch das Zertifikat indirekt in die Signatur einbezogen.

4.2.4 Dateigröße

Die Angabe der Größe der Datei in Bytes kann für die Weiterverarbeitung beim Empfänger von Bedeutung sein. Sie kann z.B. verwendet werden, um genügend Speicherplatz zu reservieren, bevor die Datei gespeichert wird.

4.2.5 Signaturdatum und -zeit

Gemäß § 4 Abs.1 Nr. 7 SigV ist bei der Prüfung digitaler Signaturen festzustellen, ob das Signaturschlüssel-Zertifikat zum Zeitpunkt der Signaturerzeugung gültig war. Durch Angabe von Signaturdatum und -zeit kann der Signierer dem Verifizierer diesen Zeitpunkt mitteilen.[66]

Der Empfänger kann sich jedoch im Allgemeinen nicht darauf verlassen, daß der angegebene Signaturzeitpunkt mit dem tatsächlichen Signaturzeitpunkt identisch ist. Der Signierer kann sich in der Regel sogar frei entscheiden, welches Datum er einträgt, bzw. durch die von ihm verwendete Applikation eintragen läßt.

Dies gilt selbst dann, wenn der Signierer (und seine Signaturkomponente) implizites Vertrauen genießten stets den tatsächlichen Signaturzeitpunkt anzugeben[67]. Der Verifizierer kann sich auch in diesem Fall auf die Angabe des Signaturzeitpunktes nur dann verlassen, wenn er sicher ist, daß der Signaturschlüssel nicht kompromittiert ist. Falls der Signaturschlüssel kompromittiert ist, muß befürchtet werden, daß die digitale Signatur unecht ist und damit auch die Angabe des Signaturzeitpunktes keinerlei Vertrauen genießt.

Falls der Empfänger sich auf die Angaben zu Signaturdatum und Zeit verläßt und deshalb die Gültigkeitsprüfung der digitalen Signatur bezogen auf den angegebenen Zeitpunkt durchführt, trägt er das Risiko, daß er die digitale Signatur akzeptiert, obwohl das Signaturschlüssel-Zertifikat zum tatsächlichen Signaturzeitpunkt gesperrt war. Im Rahmen dieser Gültigkeitsprüfung kann der Verifizierer auch durch eine Verzeichnisabfrage nicht erkennen, daß das Zertifikat zum tatsächlichen Signaturzeitpunkt möglicherweise gesperrt war.[68]

Der angegebene Signaturzeitpunkt ist daher im Allgemeinen kein geeigneter Prüfzeitpunkt bei der Gültigkeitsprüfung digitaler Signaturen. Im Allgemeinen sind nur Prüfzeitpunkte geeignet, bei denen sich der Verifizierer sicher sein kann, daß die digitale Signatur zu dem Zeitpunkt bereits existiert hat. Da der Verifizierer im Zweifel beweisen können muß, daß er bei der Gültigkeitsprüfung einen geeigneten Zeitpunkt zugrundegelegt hat, sieht das Signaturgesetz die Möglichkeit vor, einen Zeitstempel einzuholen, der als Beweismittel für die Existenz der digitalen Signatur zum im Zeitstempel ausgewiesenen Zeitpunkt dienen kann.[69]

Ohne ein solcher Beweismittel geht der Verifizierer somit stets ein Risiko ein. Der Verifizierer kann jedoch bereit sein, dieses Risiko zu tragen. Es kann darüber hinaus auch vollständig auf eine technische Gültigkeitsprüfung einer digitalen Signatur verzichten. Unter diesen Voraussetzungen kann die Angabe des Signaturzeitpunktes als Hinweis auf den Erstellungszeitpunkt verwendet werden.

---

66 Entsprechend der Datumsangabe in Papier-Dokumenten.

67 Implizites Vertrauen genießen z.B. die Dienste der ZS.

68 Bei einer Anfrage an den Verzeichnisdienst bezogen auf die Angaben zu Signaturdatum und -zeit als Abfragezeitpunkt (vgl. SigI-spezifische Erweiterung „verifyAt“ im Kapitel 2.2.1 [BSI-DIR 99]) ist die korrekte Antwort des Verzeichnisdienstes: Das Zertifikat ist nicht gesperrt.

69 Vgl. Kapitel 3.1 [BSI-TSP 99], „Schutz des Empfängers einer Nachricht“.

4.2.6 Ort

Der geographische Ort, an dem die Daten signiert wurden, kann für den Empfänger von Bedeutung sein. Entsprechend der Ortsangabe innerhalb von Dokumenten kann die Angabe des Ortes dem Empfänger einen Hinweis darauf geben, wo die Daten signiert wurden.[70]

4.2.7 Signaturnummer

Die Signaturnummer ist der Wert eines Signaturzählers des Signierers. Der Signaturzähler inkrementiert die Signaturnummer mit jeder Signatur. Durch die Einbeziehung der Signaturnummer in das Signaturaustauschobjekt kann ein Hinweis auf eine zeitliche Ordnung gegeben werden. Die Signaturnummer ermöglicht darüber hinaus bei Vorlage zweier identischer Signaturen (z.B. im Falle von Überweisungsaufträgen, Bestellungen, o.ä.) eine Unterscheidung, ob es sich um verschiedene Signaturen handelt oder um die gleiche Signatur (Original und Kopie).[71]

4.2.8 Automatisch erstellte Signatur

Ein Hinweis darauf, daß die Signatur in einem automatischen Verfahren generiert wurde, kann insbesondere dann von Bedeutung sein, wenn Nutzdaten signiert werden, die nicht vom Signierer generiert worden sind (z.B. wenn eine automatische Quittung erstellt wird).

4.2.9 Mitzeichnung

Im Falle der Mitzeichnung werden in ein Signaturaustauschobjekt mehrere digitale Signaturen (Mehrfachsignaturen) integriert. Es werden zwei Fälle der Mitzeichnung unterschieden. Bei der parallelen Mitzeichnung werden mehrere digitale Signaturen generiert; pro Mitzeichner eine. Mitzeichnungen können auch seriell erfolgen. Serielle Mitzeichnungen werden Gegenzeichnungen genannt.

Bei einer Gegenzeichnung wird eine digitale Signatur sequentiell durch den Gegenzeichner signiert.[72] In das Signaturaustauschobjekt wird neben der digitalen Signatur des Erstzeichners eine weitere digitale Signatur des Gegenzeichners integriert.

In die digitale Signatur des Erstzeichners kann die digitale Signatur des Gegenzeichners naturgemäß nicht eingeschlossen werden, da der Erstzeichner die digitale Signatur bereits generiert hat, bevor die Gegenzeichnung erfolgt. Eine nachträgliche Einbeziehung der Gegenzeichnung in die digitale Signatur des Erstzeichners ist deshalb nicht möglich.[73]

In beiden Fällen der Mitzeichnung ist es daher möglich, das Signaturaustauschobjekt unerkannt so zu manipulieren, daß dem Verifizierer die Tatsache des Vorliegens einer Gegenzeichnung oder einer weiteren parallelen Signatur verborgen bleibt. Mögliche Angriffe,

---

70 Mehr als ein Hinweis kann diese Angabe naturgemäß nicht sein (vgl. Kapitel 4.2.5).

71 Mehr als ein Hinweis kann die Angabe der Signaturnummer naturgemäß nicht sein (vgl. Kapitel 4.2.5).

72 Die Gegenzeichnung unterscheidet sich dadurch von einer Mehrfachsignatur, bei der der Inhalt parallel mehrfach signiert wird.

73 Bei parallelen Signaturen ist eine Einbeziehung der digitalen Signatur eines anderen Signierers bereits per Definition ausgeschlossen.

die auf einer solchen Manipulation beruhen, müssen auf andere nicht spezifizierte Art und Weise abgewehrt werden, sofern sie relevant sind.[74]

5 Signaturaustauschformat

Das Signaturaustauschformat wird im folgenden unter Verwendung der Spezifikationsprache ASN.1 [ITUX680] beschrieben. Die Spezifikation orientiert sich an der Spezifikation des Datentyps „SignedData“, der in Kapitel 5 [CMS 98] definiert ist.

Das Signaturaustauschformat enthält optionale Felder, die in Signaturaustauschobjekten vorhanden sein können oder nicht. In der vorliegenden Spezifikation wird durch die Forderung nach Unterstützung eines Feldes zum Ausdruck gebracht, daß die Applikation es erkennen und auswerten bzw. generieren können muß.

Alle Felder müssen unterstützt werden, sofern nicht explizit angegeben wird, daß eine Unterstützung nicht gefordert wird. Gefordert wird allerdings, daß Applikation nicht zu unterstützende Felder ignorieren, falls sie sie nicht auswerten können oder wollen.

Die allgemeine Syntax für kryptographische Nachrichten ist gemäß Kapitel 2 [CMS 98]:

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

Die kryptographische Nachricht assoziiert den zu schützenden Inhalt (content) mit einer Typangabe (contentType) in Form eines OIDs.

Das Signaturaustauschformat wird mit folgendem OID assoziiert, der es eindeutig identifiziert.

```
id-signedData OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                                      rsadsi(113549) pkcs7(7) 2 }
```

[74] Dies kann z.B. dadurch erfolgen, daß den Nutzdaten eine Mitzeichnungsliste beigefügt ist. Der Empfänger kann dann anhand dieser Liste prüfen, ob das erhaltene Signaturaustauschobjekt alle Signaturen enthält, die es entsprechend der Liste enthalten müßte.

Dieser OID identifiziert den Datentyp „SignedData“, dessen Struktur in folgender Abbildung im Überblick dargestellt wird:

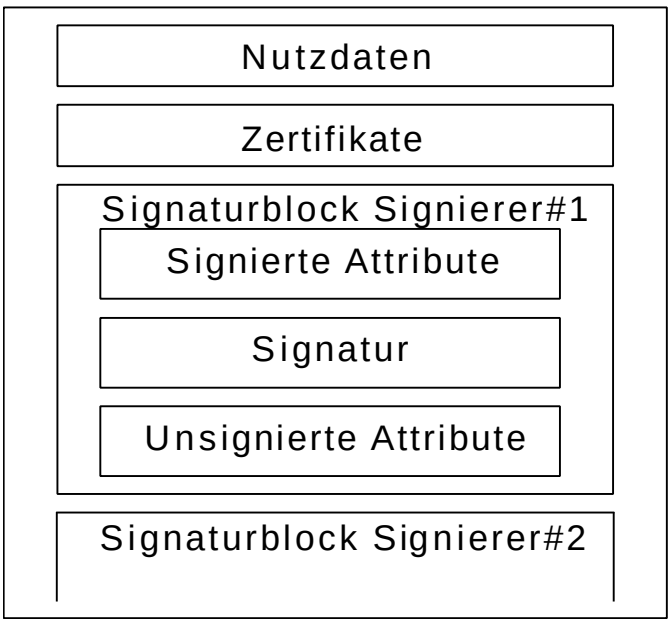

Abbildung 1: Signaturaustauschformat

Für den Datentyp „SignedData“ wird folgende Syntax verwendet:

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

5.1 Version

Das Versionsfeld (version) bezeichnet die Version der Syntax. Es sind folgende Werte definiert:

```
CMSVersion ::= INTEGER  { v0(0), v1(1), v2(2), v3(3), v4(4) }
```

Für den Datentyp „SignedData“ muß in Übereinstimmung mit [CMS 98] stets einer der beiden Identifier „v1“ mit dem Integer-Wert „1“ oder Identifier „v3“ mit dem Integer-Wert „3“ verwendet werden. Der Identifier „v1“ ist zu verwenden, wenn keine Attribut-Zertifikate Teil des Signaturaustauschobjekts sind und der Typ der Nutzdaten „id-data“ ist.[75] Andernfalls ist stets der Identifier „v3“ zu verwenden.

[75] Durch den OID „id-data“ wird der Datentyp „beliebige binäre Daten“ identifiziert (vgl. Kapitel 4.1.1). Im Anhang A der vorliegenden Spezifikation werden weitere OIDs zur exakteren Identifizierung von Datentypen identifiziert.

5.2 Hashverfahren

Das Feld für die Hash-Algorithmen (digestAlgorithms) dient ausschließlich dem Zweck, die Verifikation signierter Daten in einem „Durchgang“ zu ermöglichen.[76] Es werden folgende Datenstrukturen verwendet:

```
DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier

DigestAlgorithmIdentifier  ::= AlgorithmIdentifier

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

Da die Syntax Mehrfachsignaturen vorsieht, können mehrere Signierer unterschiedliche Hashverfahren verwenden. Alle verwendeten Hashverfahren bilden die Menge, die in dem Feld für die Hash-Algorithmen einzutragen ist.[77]

5.3 Nutzdaten

Für das Inhaltsfeld (encapContentInfo) wird folgende Syntax verwendet:

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

     ContentType        ::= OBJECT IDENTIFIER
```

Das Feld „eContentType“ enthält den OID zur eindeutigen Identifizierung des Inhaltes[78]. Das Feld „eContent“ enthält in der Regel den Inhalt, d.h. die Nutzdaten, in Form eines OCTET STRING. Es wird nicht gefordert, daß der Inhalt DER-kodiert[79] ist.

Ausnahmsweise ist dieses Feld nicht Teil des Signaturaustauschobjektes, wenn eine externe Signatur gebildet wird.[80] Die Nutzdaten sind dann z.B. in einer anderen Teilnachricht oder einer unabhängig von der Nachricht zu übermittelnden Datei enthalten.

---

76 Während der Auflösung der Datenstruktur können bereits die Hashwerte gebildet werden, die später bei der Verifikation benötigt werden.

77 **Achtung**: Diese Forderung ist in [CMS 98] nicht enthalten. Dort wird lediglich die Intention des Feldes angegeben, allen Verifizierern die Prüfung in einem Durchgang zu ermöglichen, falls bei Mehrfachsignaturen unterschiedliche Hashverfahren verwendet werden.

Insbesondere bei digitalen Signaturen umfangreicher Nachrichten kann es den erforderlichen Zeitaufwand für die Prüfung nahezu halbieren, wenn alle verwendeten Hashverfahren in dem Feld angegeben werden. Da diesem Vorteil keine wesentlichen Nachteile gegenüber stehen, erscheint die Forderung gerechtfertigt.

78 Durch den OID „id-data“ wird der Datentyp „beliebige binäre Daten“ identifiziert (vgl. Kapitel 4.1.1). Im Anhang A der vorliegenden Spezifikation werden weitere OIDs zur exakteren Identifizierung von Datentypen identifiziert.

79 Zur DER-Kodierung vgl. [ITUX690 94].

80 Deshalb ist das Feld in der Syntax als optional gekennzeichnet. Dies bedeutet jedoch nicht, daß das Feld nicht unterstützt werden müßte.

5.4 Zertifikate

Das optionale Zertifikatsfeld (certificates) enthält eine Menge von Zertifikaten. Eine Notwendigkeit für die Verwendung dieses Feldes besteht nicht. Das gilt auch dann, wenn Zertifikate Beschränkungen nach § 7 Abs.1 Nr.7 oder § 7 Abs. 2 SigG enthalten oder wenn Zertifikate nicht abrufbar sind. Diese Zertifikate sind in die Signatur einzuschließen und sind deshalb als signiertes Attribut Teil der Datenstruktur „SignerInfo“ (s.u.). Der Verifizierer enthält die Zertifikate des Zertifizierungspfades stets als signiertes Attribut oder kann sie aus Zertifikatsverzeichnissen abrufen.

Im letztgenannten Fall könnte das Feld verwendet werden, um dem Verifizierer dem Abruf von Zertifikaten zu ersparen. Sinnvoll ist dies jedoch nur dann, wenn der Verifizierer auf Abfragen des Verzeichnisdienstes voraussichtlich verzichten wird. Andernfalls wird der Verifizierer das Feld ohnehin nicht verwenden.

Falls das Feld verwendet wird, sollte es für alle Signierer alle Signaturschlüssel-Zertifikate des Zertifizierungspfades und die für die Verifizierung erforderlichen Attribut-Zertifikate enthalten.

Das Feld muß nicht unterstützt werden.

5.5 Signaturblock

Das Informationsfeld (signerInfos) enthält eine Menge von Informationen über den oder die Signierer. Es wird folgende Syntax aus Kapitel 5.3 [CMS 98] verwendet:

```
SignerInfos             ::= SET OF SignerInfo

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

Die Syntax sieht Mehrfachsignaturen vor. Das Feld „signerInfo“ kann deshalb mehrfach vorkommen.

Das Versionsfeld (version) gibt die Version der Syntax für die Datenstruktur „SignerInfo“ an.[81] Entsprechend [CMS 98] ist stets der Identifier „v1“ mit dem Integer-Wert „1“ zu verwenden.

Das Feld „issuerAndSerialNumber“ identifiziert das Signaturschlüssel-Zertifikat und damit den Signaturschlüssel, mit dem die Daten signiert wurden. Da dieses Feld nicht in die Signatur eingeschlossen wird, ist es vom Verifizierer nicht zu verwenden. Verwendet werden muß statt dessen das signierte Attribut vom Typ „AttrRef“, das den gleichen Inhalt hat, bzw. das signierte Attribut „Certificate“, das das Signaturschlüssel-Zertifikat enthält (s.u. Feld „signedAttrs“ und Kapitel 6.1.3). Das Feld „issuerAndSerialNumber“ ist somit an sich überflüssig, darf jedoch nicht ausgelassen werden, da es nicht optional ist.

Es wird folgende Datenstruktur verwendet:[82]

---

81 Die Datenstruktur „CMSVersion“ wurde bereits oben definiert.

```
IssuerAndSerialNumber     ::= SEQUENCE {
     issuer                   Name,
     serialNumber             CertificateSerialNumber }
```

Das Ausstellerfeld (issuer) enthält den technischen Namen der ZS wie in Kapitel 2.3.4 [BSI-ZERT 99] beschrieben. Das Seriennummernfeld (serialNumber) enthält die Seriennummer wie in Kapitel 2.3.2 [BSI-ZERT 99] beschrieben.

Das Feld „digestAlgorithm“ enthält den OID für die verwendete Hashfunktion. Die Hashfunktion muß auch im Feld „digestAlgorithms“ enthalten sein, das in Kapitel 5.2 beschrieben ist.

Da dieses Feld nicht in die Signatur eingeschlossen wird, ist es vom Verifizierer nicht zu verwenden. Verwendet werden muß statt dessen die Angabe über die mit dem öffentlichen Signaturschlüssel zu verwendende Hashfunktion im Signaturschlüssel-Zertifikat des Signierers bzw. im Padding-String (vgl. Kapitel 4.1.7).

Das Feld „digestAlgorithm“ ist somit an sich überflüssig, darf jedoch nicht ausgelassen werden, da es nicht optional ist.

Das Feld „signedAttrs“ enthält die Menge aller Attribute, die in die Signatur einbezogen werden. Es wird folgende Syntax verwendet:

```
SignedAttributes ::= SET SIZE (1..MAX) OF Attribute

Attribute        ::= SEQUENCE {
     attrType         OBJECT IDENTIFIER,
     attrValues       SET OF AttributeValue }

AttributeValue ::= ANY
```

Jedes Attribut wird durch einen OID und einen Wert bestimmt, dessen Typ durch den OID festgelegt wird. Die Attributtypen sind in Kapitel 6 spezifiziert. Das Feld ist nicht optional.[83]

---

82 Statt diesem Feld wurde in die aktuelle Version von CMS das Feld „sid“ aufgenommen (s. Kapitel 5.3 [CMS98 V10]). Es wird folgende Syntax verwendet:

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

Anstatt durch den Namen der ausstellenden CA und die Seriennummer des Zertifikats könnte der Signaturschlüssel entsprechend dieser Syntax auch durch einen Identifizierer für den Schlüssel, den „SubjectKeyIdentifier“, bestimmt werden.

Diese Syntax wird nicht verwendet, weil sie nicht der Version 2 der MailTrusT-Spezifikation entspricht (s. Kapitel 9.3.1 [TTT-AF 99]) und von der Auswahlmöglichkeit ohnehin kein Gebrauch gemacht werden darf.

Da die CMS z.Z. noch in Bearbeitung ist, muß ein bestimmter Stand als Basis für die vorliegende Spezifikation gewählt werden. Der Stand muß für SigI und MTT identisch sein, um unnötigen Implementierungsaufwand zu vermeiden (s. Abschnitt B3 der Spezifikation, „SigG-Erweiterungen zu MailTrusT-Spezifikation“).

Hinzu kommt, daß Zertifikate stets durch den technischen Namen der ZS und die Seriennummer des Zertifikates eindeutig identifiziert werden. Die Zertifikatserweiterung „SubjectKeyIdentifier“ muß bei der Erstellung von Zertifikaten nicht unterstützt werden. (s. Kapitel 2.3.9.8 [BSI-ZERT99]).

83 In [CMS 98] wird dieses Feld als optional angesehen, da es dem Aussteller einer digitalen Signatur freigestellt wird, Attribute in die digitale Signatur einzubeziehen. Für die vorliegende Spezifikation gilt dies nicht, da einige Attribute stets in die Signatur einzubeziehen sind.

Das Feld „signatureAlgorithm“ identifiziert den Signatur-Algorithmus und dessen Parameter, soweit vorhanden. Da dieses Feld nicht in die Signatur eingeschlossen wird, ist es vom Verifizierer nicht zu verwenden.

Der Verifizierer muß den verwendeten Signatur-Algorithmus statt dessen aus dem Feld „SubjectPublicKeyInfo“ des Signaturschlüssel-Zertifikates entnehmen (vgl. Kapitel 4.1.6). Das Zertifikat bzw. eine eindeutige Referenz auf das Zertifikat muß stets in einem signierten Attribut vom Typ „Certificate“ bzw. „RefCert“ enthalten sein (s.o. Feld „signedAttrs“ und Kapitel 6.1.3 bzw. 6.1.4). Das Feld „singnatureAlgorithm“ ist somit an sich überflüssig, darf jedoch nicht ausgelassen werden, da es nicht optional ist.

Es wird folgende Datenstruktur verwendet:

```
SignatureAlgorithmIdentifier ::= AlgorithmIdentifier
```
[84]

Das Signaturfeld (signature) enthält die Signatur als Ergebnis der Anwendung des Signatur-Algorithmus und des privaten Schlüssels auf die zu signierenden Daten. Es wird folgende Datenstruktur verwendet:

```
SignatureValue ::= OCTET STRING
```

Die Signaturbildung wird in Kapitel 7 beschrieben.

Das Feld „unsignedAttrs“ enthält Attribute, die nicht in die Signatur einbezogen werden. Im übrigen gilt das gleiche wie für das Feld „signedAttrs“. Das Feld wird nur beim Attributtyp Gegenzeichnung verwendet (s. Kapitel 6.2.9).

Das optionale Feld „unsignedAttrs“ muß nicht unterstützt werden.

Wie beschrieben existieren für die meisten Felder des Signaturblocks entsprechende signierte Attribute, die bei der Verifikation stets zu verwenden sind, da sie durch die digitale Signatur geschützt sind. Signaturgesetz-konforme Applikationen müssen die geschützten Felder auswerten.

Applikationen, die nicht der vorliegenden Spezifikation entsprechen, werden bei der Verifikation statt der geschützten Felder die ungeschützten Felder des Signaturblocks verwenden. Da es einem Verifizierer unbenommen bleibt, auf eigenes Risiko auch nicht Signaturgesetz-konforme Applikationen zu verwenden, müssen die Inhalte der geschützten und der ungeschützten Felder jeweils übereinstimmen.

6 Attributtypen

Für die in Kapitel 4 definierten Attribute digital signierter Daten werden in diesem Kapitel Attributtypen spezifiziert. Bis auf den Attributtyp Gegenzeichnung müssen alle Attribute in die Signatur einbezogen werden. Von der in [CMS 98] prinzipiell eingeräumten Wahlmöglichkeit hinsichtlich der Einbeziehung optionaler Attribute darf kein Gebrauch gemacht werden. Es gilt der Grundsatz, daß alle Attribute in die Signatur einbezogen werden müssen, die einbezogen werden können.[85]

---

[84] Die Datenstruktur „AlgorithmIdentifier“ ist in Kapitel 5.2 definiert.

[85] Dies stellt eine Vereinfachung zugunsten der Sicherheit der Signatur dar. Der Signierer könnte durch eine Wahlfreiheit bei der Einbeziehung von Attributen Fehler begehen, die für ihn ungünstige Folgen haben könnten. Der Verifizierer könnte z.B. fälschlicherweise davon ausgehen, daß alle Attribute in die Signatur einbezogen sind und deshalb einzelnen Attributen ein ungerechtfertigtes Vertrauen entgegenbringen.

6.1 Obligatorisch zu unterstützende Attribute

Die folgenden Attribute müssen Teil eines jeden Signaturblocks sein und sind stets in die digitale Signatur einbezogen. Alle Attribute müssen von allen Applikationen unterstützt werden, d.h. sowohl generiert als auch ausgewertet werden können.

6.1.1 Typ (Content Type)

Das Attribut vom Typ „Content Type“ spezifiziert den Typ der Nutzdaten. Das Attribut wird durch folgenden OID identifiziert:

```
id-contentType OBJECT IDENTIFIER ::= { iso(1) member-body(2)
                        us(840) rsadsi(113549) pkcs(1) pkcs9(9) 3 }
```

Die Attributwerte haben den Typ „ContentTyp“, der wie folgt definiert ist:

```
ContentType ::= OBJECT IDENTIFIER
```

Der Attributwert referenziert den Typ der signierten Daten. Die verschiedenen Typen sind in Anhang A aufgelistet.

Auch wenn die verwendete Syntax für das Feld „attrValues“ die Angabe einer Menge von Attributwerten zulassen würde, darf immer nur genau ein Wert angegeben werden. Außerdem darf dieses Attribut in einem Signaturblock immer nur einmal enthalten sein, d.h. mehrere Instanzen des Attributes sind verboten.

6.1.2 Hashwert (Message Digest)

Das Attribut vom Typ „Message Digest“ spezifiziert den Hashwert über den OCTET STRING, der signiert wird (vgl. Teilfeld „encapContentInfo“ der Datenstruktur „SignedData“ in Kapitel 5.3).

Bei externen Signaturen bleibt das Teilfeld „encapContentInfo“ leer. In diesem Fall wird der Hashwert so gebildet, als wenn die Nutzdaten in diesem Feld enthalten wären.

Das Attribut muß aus technischen Gründen, die mit der Signaturbildung zusammen hängen, stets in den signierten Daten enthalten sein (s. Kapitel 7). Die Nutzdaten werden mittels dieses Attributes in die Signatur einbezogen.

Das Attribut wird durch folgenden OID identifiziert:

```
id-messageDigest OBJECT IDENTIFIER ::= { iso(1) member-body(2)
                        us(840) rsadsi(113549) pkcs(1) pkcs9(9) 4 }
```

Attributwerte haben den Typ „MessageDigest“, der wird wie folgt definiert ist:

```
MessageDigest ::= OCTET STRING
```

Auch wenn die verwendete Syntax für das Feld „attrValues“ die Angabe einer Menge von Attributwerten zulassen würde, darf immer nur genau ein Wert angegeben werden. Außerdem darf dieses Attribut in einem Signaturblock immer nur einmal enthalten sein, d.h. mehrere Instanzen des Attributes sind verboten.

Auf der anderen Seite entstehen für den Signierer dadurch keine wesentlichen Nachteile, daß er die Wahlfreiheit nach [CMS 98] nicht hat.

6.1.3 Signaturschlüssel-Zertifikat des Signierers

Ein Attribut vom Typ „Certificate“ spezifiziert das Signaturschlüssel-Zertifikat des Signierers. Das Attribut wird im Rahmen dieser Spezifikation neu definiert und durch folgenden OID identifiziert:

```
id-sigi                    OBJECT IDENTIFIER ::= { 1 3 36 8 }
id-sigi-sigAttrs           OBJECT IDENTIFIER ::= { 1 3 36 8 X[86] }
id-sigi-sigAttrs-cert      OBJECT IDENTIFIER ::= { 1 3 36 8 X 1 }
```

Attributwerte haben den Typ „Certificate“, der in Kapitel 2 [BSI-ZERT 99] definiert ist. Es gilt also:

```
Sigi-SigAttrs-Cert      ::= Certificate
```

Auch wenn die verwendete Syntax für das Feld „attrValues“ die Angabe einer Menge von Attributwerten zulassen würde, darf immer nur genau ein Wert angegeben werden. Außerdem darf dieses Attribut in einem Signaturblock immer nur einmal enthalten sein, d.h. mehrere Instanzen des Attributes sind verboten.[87]

In jedem Signaturblock darf alternativ immer nur ein Attribut vom Typ „Certificate“ oder vom im folgenden definierten Typ „CertRef“ enthalten sein, der eine Referenz auf das Signaturschlüssel-Zertifikat spezifiziert.

6.1.4 Referenz auf Signaturschlüssel-Zertifikat des Signierers

Ein Attribut vom Typ „CertRef“ spezifiziert eine Referenz auf das Signaturschlüssel-Zertifikat des Signierers. Die Referenz wird benötigt, um das Zertifikat aus dem Zertifikatsverzeichnis abrufen zu können.

Anfragen an den Verzeichnisdienst sind in Kapitel 2.2.1 [BSI-DIR 99] beschrieben. Zertifikate werden durch die Datenstruktur „CertID“ identifiziert. Zur eindeutigen Identifizierung des Zertifikates genügt stets das Wissen über den Namen des Ausstellers des Zertifikates und die Seriennummer des Zertifikates.

Mit diesem Informationen können Zertifikate abgerufen werden. Dabei ist zu beachten, daß beim Abruf statt des Namens des Ausstellers der Zertifikates ein Hashwert über den Namen verwendet. Da in der Referenz stets der X.500-Distinguished-Name des Ausstellers angegeben wird, muß somit vor einem Abruf noch eine Hashwertbildung erfolgen.

Das Attribut wird im Rahmen dieser Spezifikation neu definiert und wird durch folgenden OID identifiziert:

```
id-sigi-sigAttrs-certRef  OBJECT IDENTIFIER ::= { 1 3 36 8 X 2 }
```

Attributwerte haben den Typ „IssuerAndSerialNumber“ der in Kapitel 5.5 definiert ist. Es gilt also:

```
Sigi-SigAttrs-CertRef   ::= IssuerAndSerialNumber
```

---

86 Der Wert für „sigattrs“ ist noch festzulegen.

87 Andere Zertifikate dürfen daher nur über das Feld „certificates“ in das Signaturaustauschobjekt einbezogen werden. Die Notwendigkeit, den gesamten Zeritifizierungspfad in die Signatur einzubeziehen besteht nicht, da Zertifizierungspfade im Rahmen der vorliegenden Spezifikation stets eindeutig sind.

Auch wenn die verwendete Syntax für das Feld „attrValues“ die Angabe einer Menge von Attributwerten zulassen würde, darf immer nur genau ein Wert angegeben werden. Außerdem darf dieses Attribut in einem Signaturblock immer nur einmal enthalten sein, d.h. mehrere Instanzen des Attributes sind verboten.

Jeder Signaturblock darf alternativ immer nur ein Attribut vom Typ „Certificate“ oder vom Typ „CertRef“ enthalten.

6.1.5 Attribut-Zertifikate des Signierers

Ein Attribut vom Typ „AttibuteCertificate“ spezifiziert ein oder mehrere Attribut-Zertifikate. Das Attribut wird im Rahmen dieser Spezifikation neu definiert und durch folgenden OID identifiziert:

```
id-sigi-sigAttrs-attrCert  OBJECT IDENTIFIER ::= { 1 3 36 8 X 3 }
```

Attributwerte haben den Typ „AttributeCertificate“, der in Kapitel 3.1 [BSI-ZERT 99] definiert ist. Es gilt also:

```
Sigi-SigAttrs-AttrCert  ::= AttributeCertificate
```

Falls der Signierer mehrere Attribut-Zertifikate in einen Signaturblock einfügen will, so muß er (bzw. sein Applikation) alle diese Zertifikate in einem Attribut dieses Typs zusammenfassen. Es darf somit in einem Signaturblock nur ein Attribut vom Typ „AttributeCertifikate“ geben, das allerdings mehrere Werte, d.h. mehrere Attribut-Zertifikate haben kann. Mehrere Instanzen des Attributs sind jedoch auch hier verboten.

Bei mehreren Attribut-Zertifikaten ist es zulässig, teilweise die Attribut-Zertifikate und teilweise Referenzen auf Attribut-Zertifikate zu verwenden. Neben einem Attribut vom Typ „AttributeCertificate“ kann es somit ein weiteres Attribut vom Typ „AttrRef“ geben, der im folgenden definiert wird. Jedes Attribut-Zertifikat darf jedoch nur einmal unter Verwendung einer dieser alternativen Formen im Signaturblock vorhanden sein.

6.1.6 Referenz auf Attribut-Zertifikat des Signierers

Ein Attribut vom Typ „AttrRef“ spezifiziert eine Referenz auf ein Attribut-Zertifikat des Signierers. Das Attribut wird im Rahmen dieser Spezifikation neu definiert und wird durch folgenden OID identifiziert:

```
id-sigi-sigAttrs-attrRef  OBJECT IDENTIFIER ::= { 1 3 36 8 X 4 }
```

Attributwerte haben den Typ „IssuerAndSerialNumber“ der in Kapitel 5.5 definiert ist. Es gilt also:

```
Sigi-sigAttrs-AttrRef   ::= IssuerAndSerialNumber
```

Falls der Signierer mehrere Referenzen auf Attribut-Zertifikate in einen Signaturblock einfügen will, so muß er (bzw. sein Applikation) alle diese Referenzen in einem Attribut dieses Typs zusammenfassen. Es darf somit in einem Signaturblock nur ein Attribut vom Typ „AttrRef“ geben, das allerdings mehrere Werte, d.h. mehrere Referenzen auf Attribut-Zertifikate haben kann. Mehrere Instanzen des Attributs sind jedoch auch hier verboten.

Bei mehreren Attribut-Zertifikaten ist es zulässig, teilweise die Attribut-Zertifikate und teilweise Referenzen auf Attribut-Zertifikate zu verwenden. Neben einem Attribut vom Typ „AttrRef“ kann es somit ein weiteres Attribut vom Typ „AttributeCertificate“ geben. Jedes Attribut-Zertifikat darf

jedoch nur einmal unter Verwendung einer dieser alternativen Formen im Signaturblock vorhanden sein.

6.2 Optional zu unterstützende Attribute

Die folgenden Attribute können optional verwendet werden. Alle Attribute mit Ausnahme des Attributs für die Gegenzeichnung müssen in die Signatur einbezogen werden.

Die folgenden Attribute können Teil eines jeden Signaturblocks sein, um an die eigentlichen Nutzdaten weitere nützliche Informationen zu binden. Es besteht jedoch keine Verpflichtung, die Attribute zu verwenden. Die Verwendung der Attribute muß von Applikationen nicht unterstützt werden, da sich eine entsprechende Forderung aus SigG/SigV nicht ableiten läßt.

Falls die Attribute verwendet werden, besteht somit keine Garantie, daß sie von den Empfängern ausgewertet werden können. Applikationen müssen die Attribute nicht auswerten. Sie müssen jedoch alle Attribute ignorieren, die sie nicht auswerten können, und dem Empfänger anzeigen, daß nicht auswertbare Attribute vorhanden sind.

6.2.1 Quittungsanforderung (Receipt Request)

Ein Attribut vom Typ „ReceiptRequest“ spezifiziert eine Quittungsanforderung für die signierten Daten. Das Attribut ist in [ESS 99] definiert.

Anmerkung: Die Spezifikation des Attributs und des zugehörigen komplexen Quittungsverfahrens wird zunächst zurückgestellt. Bei Bedarf kann die Spezifikation in Teil B der SigI erfolgen.

6.2.2 Dateibezeichner (File Name)

Ein Attribut vom Typ „FileName“ spezifiziert einen Dateibezeichner, der insbesondere bei externen Signaturen von Bedeutung ist (vgl. Kapitel 4.2.2).[88] Das Attribut wird im Rahmen dieser Spezifikation neu definiert und durch folgenden OID identifiziert:

```
id-sigi-sigAttrs-fileName     OBJECT IDENTIFIER ::= { 1 3 36 8 X 5 }
```

Die Attributwerte haben den Typ „IA5String“. Es gilt also:

```
Sigi-SigAttrs-FileName    ::= IA5String
```

Auch wenn die verwendete Syntax für das Feld „attrValues“ die Angabe einer Menge von Attributwerten zulassen würde, darf immer nur genau ein Wert angegeben werden. Außerdem darf dieses Attribut in einem Signaturblock immer nur einmal enthalten sein, d.h. mehrere Instanzen des Attributes sind verboten. Den Nutzdaten darf somit immer nur ein Dateibezeichner zugeordnet werden.

6.2.3 Speicherzeitpunkt (Storage Time)

Ein Attribut vom Typ „Storage Time“ spezifiziert den Speicherzeitpunkt einer Datei (vgl. Kapitel 4.2.3).[89] Das Attribut wird im Rahmen dieser Spezifikation neu definiert und wird durch folgenden OID identifiziert:

---

88 Falls als Austauschformat MIME verwendet wird (vgl. Kapitel 8), kann statt des Attributs auch ein Parameter in der Kopfzeile einer S/MIME-Nachricht verwendet werden. Der Parameter „filename-parm“ des sog. „Content-Disposition Headers“ kann dem gleichen Zweck dienen (vgl. [RFC 2183 97]. Dieser Parameters wird jedoch nicht in die digitale Signatur einbezogen.

```
id-sigi-sigAttrs-storageTime  OBJECT IDENTIFIER ::= { 1 3 36 8 X 6 }
```

Die Attributwerte haben den Typ „Time“, der in Kapitel 2.3.5 [BSI-Zert99] definiert ist. Es gilt also:

```
Sigi-SigAttrs-StorageTime    ::= Time
```

Es ist stets „GeneralizedTime“ in dem dort angegebenen Format zu verwenden.

Auch wenn die verwendete Syntax für das Feld „attrValues“ die Angabe einer Menge von Attributwerten zulassen würde, darf immer nur genau ein Wert angegeben werden. Außerdem darf dieses Attribut in einem Signaturblock immer nur einmal enthalten sein, d.h. mehrere Instanzen des Attributes sind verboten. Den Nutzdaten darf somit immer nur ein Speicherzeitpunkt zugeordnet werden.

6.2.4 Dateigröße (File Size)

Ein Attribut vom Typ „File Size“ spezifiziert die Größe der Datei mit den Nutzdaten (vgl. Kapitel 4.2.4).[90] Das Attribut wird im Rahmen dieser Spezifikation neu definiert und wird durch folgenden OID identifiziert:

```
id-sigi-sigAttrs-fileSize  OBJECT IDENTIFIER ::= { 1 3 36 8 X 7 }
```

Die Attributwerte haben den Typ „INTEGER“. Es gilt also:

```
Sigi-SigAttrs-FileSize    ::= INTEGER
```

Auch wenn die verwendete Syntax für das Feld „attrValues“ die Angabe einer Menge von Attributwerten zulassen würde, darf immer nur genau ein Wert angegeben werden. Außerdem darf dieses Attribut in einem Signaturblock immer nur einmal enthalten sein, d.h. mehrere Instanzen des Attributes sind verboten. Den Nutzdaten darf somit immer nur ein Dateigröße zugeordnet werden.

6.2.5 Signaturzeitpunkt (Signing Time)

Das Attribut vom Typ „SigningTime“ spezifiziert den Zeitpunkt, zu dem der Signierer den Signaturprozeß durchgeführt hat (vgl. Kapitel 4.2.5).[91]. Das Attribut wird durch folgenden OID identifiziert:

```
id-signingTime OBJECT IDENTIFIER ::= { iso(1) member-body(2)
                        us(840) rsadsi(113549) pkcs(1) pkcs9(9) 5 }
```

---

89 Falls als Austauschformat MIME verwendet wird (vgl. Kapitel 8), kann statt des Attributs auch ein Parameter in der Kopfzeile einer S/MIME-Nachricht verwendet werden. Der Parameter „creation-date-parm“ des sog. „Content-Disposition Headers“ kann dem gleichen Zweck dienen (vgl. [RFC 2183 97]. Dieser Parameters wird jedoch nicht in die digitale Signatur einbezogen.

90 Falls als Austauschformat MIME verwendet wird (vgl. Kapitel 8), kann statt des Attributs auch ein Parameter in der Kopfzeile einer S/MIME-Nachricht verwendet werden. Der Parameter „size-parm“ des sog. „Content-Disposition Headers“ kann dem gleichen Zweck dienen (vgl. [RFC 2183 97]. Dieser Parameters wird jedoch nicht in die digitale Signatur einbezogen.

91 Es gibt keine Anforderung hinsichtlich der Korrektheit der Angabe des Signaturzeitpunktes. Dementsprechend bleibt es dem Empfänger der Signatur freigestellt, den angegebenen Zeitpunkt als Signaturzeitpunkt zu akzeptieren und bei der Verifikation zu verwenden. Allerdings ist davon auszugehen, daß einige Signierer wie z.B. die Dienste der ZS implizites Vertrauen genießen, stets die korrekte Zeit zu verwenden.

Die Attributwerte haben den Typ „Time“, der in Kapitel 2.3.5 [BSI-ZERT 99] definiert ist. Es gilt also:

```
SigningTime     ::= Time
```

Es ist stets „GeneralizedTime“ in dem dort angegebenen Format zu verwenden.[92]

Auch wenn die verwendete Syntax für das Feld „attrValues“ die Angabe einer Menge von Attributwerten zulassen würde, darf immer nur genau ein Wert angegeben werden. Außerdem darf dieses Attribut in einem Signaturblock immer nur einmal enthalten sein, d.h. mehrere Instanzen des Attributes sind verboten. Der Signatur darf somit immer nur ein Signaturzeitpunkt zugeordnet werden.

6.2.6 Ort (Location)

Ein Attribut vom Typ „Location“ spezifiziert den Ort, an dem die Daten signiert wurden (vgl. Kapitel 4.2.6). Das Attribut wird im Rahmen dieser Spezifikation neu definiert und wird durch folgenden OID identifiziert:

```
id-sigi-sigAttrs-location  OBJECT IDENTIFIER ::= { 1 3 36 8 X 8 }
```

Die Attributwerte haben den Typ „IA5String“. Es gilt also:

```
Sigi-SigAttrs-Location     ::= IA5String
```

Auch wenn die verwendete Syntax für das Feld „attrValues“ die Angabe einer Menge von Attributwerten zulassen würde, darf immer nur genau ein Wert angegeben werden. Außerdem darf dieses Attribut in einem Signaturblock immer nur einmal enthalten sein, d.h. mehrere Instanzen des Attributes sind verboten. Der Signatur darf somit immer nur eine Ortsangabe zugeordnet werden.

6.2.7 Signaturnummer (Signature Number)

Ein Attribut vom Typ „Signature Number“ spezifiziert den Wert eines Signaturzählers (vgl. Kapitel 4.2.7). Das Attribut wird im Rahmen dieser Spezifikation neu definiert und wird durch folgenden OID identifiziert:

```
id-sigi-sigAttrs-sigNumber  OBJECT IDENTIFIER ::= { 1 3 36 8 X 9 }
```

Die Attributwerte haben den Typ „INTEGER“. Es gilt also:

```
Sigi-SigAttrs-SigNumber     ::= INTEGER
```

Auch wenn die verwendete Syntax für das Feld „attrValues“ die Angabe einer Menge von Attributwerten zulassen würde, darf immer nur genau ein Wert angegeben werden. Außerdem darf dieses Attribut in einem Signaturblock immer nur einmal enthalten sein, d.h. mehrere Instanzen des Attributes sind verboten. Der Signatur darf somit immer nur eine Signaturnummer zugeordnet werden.

6.2.8 Automatisch erstellte Signatur (Automaticly Generated)

Ein Attribut vom Typ „Automaticly Generated“ spezifiziert die Art, wie das Dokument signiert wurde, d.h. ob die Signatur automatisch oder manuell generiert wurde (vgl. Kapitel 4.2.8). Das

---

92 Das in Kapitel 11.3 [CMS 98] angegebene Zeitformat darf nicht verwendet werden.

Attribut wird im Rahmen dieser Spezifikation neu definiert und wird durch folgenden OID identifiziert:

```
id-sigi-sigAttrs-autoGen  OBJECT IDENTIFIER ::= { 1 3 36 8 X 10 }
```

Die Attributwerte haben den Typ „BOOLEAN“. Es gilt also:

```
Sigi-SigAttrs-autoGen     ::= BOOLEAN
```

Auch wenn die verwendete Syntax für das Feld „attrValues“ die Angabe einer Menge von Attributwerten zulassen würde, darf immer nur genau ein Wert angegeben werden. Außerdem darf dieses Attribut in einem Signaturblock immer nur einmal enthalten sein, d.h. mehrere Instanzen des Attributes sind verboten. Der Signatur darf somit immer nur eine Angabe über die Art der Generierung zugeordnet werden.

6.2.9 Gegenzeichnung (CounterSignature)

Ein Attribut vom Typ „CounterSignature“ spezifiziert eine oder mehrere Signaturen des DER-kodierten Wertes des Feldes „signature“ der Datenstruktur „SignerInfo“. Dadurch wird eine Signatur seriell noch einmal signiert (vgl. Kapitel 4.2.9).[93]

Im Unterschied zu allen anderen Attributen muß ein Attribut vom Typ Gegenzeichnung naturgemäß stets ein unsigniertes Attribut sein, da die Gegenzeichnung erst erfolgt, nachdem die Signatur bereits gebildet wurde. Nur als unsigniertes Attribut kann sie später noch den signierten Daten hinzugefügt werden, ohne die Signatur zu verfälschen.

Folgende Abbildung gibt den Aufbau einer Signaturaustauschobjektes mit Attributen vom Typ Gegenzeichnung wieder, wobei der Übersichtlichkeit halber nur die Singnaturblöcke dargestellt sind:

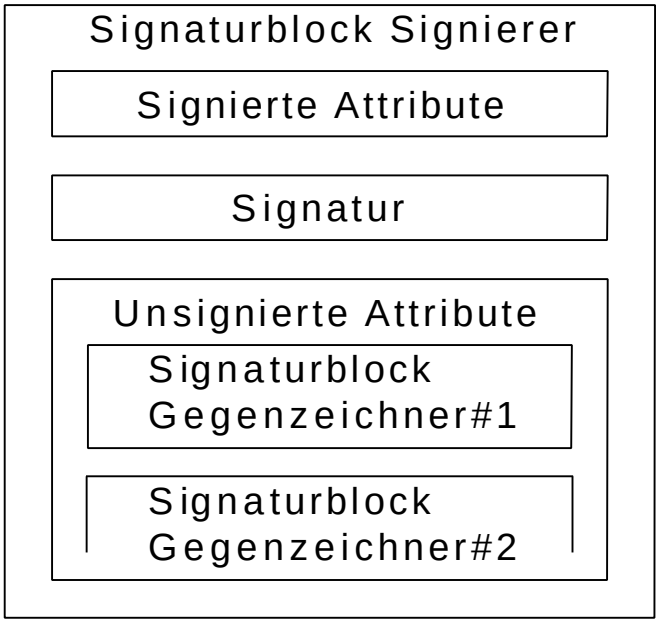

Abbildung 2: Gegenzeichnung

Ein Attribut vom Typ Gegenzeichnung wird durch folgenden OID identifiziert:

```
id-countersignature OBJECT IDENTIFIER ::= { iso(1) member-body(2)
```

93 Die Gegenzeichnung unterscheidet sich dadurch von einer Mehrfachsignatur, bei der Inhalt parallel mehrfach signiert wird.

```
                    us(840) rsadsi(113549) pkcs(1) pkcs9(9) 6 }
```

Die Attributwerte haben den Typ „SignerInfo“, der bereits in Kapitel 5.5 beschrieben ist. Es gilt also:

```
Countersignature    ::= SignerInfo
```

Die Signaturblöcke für Gegenzeichnungen entsprechen denen für normale Signaturen mit folgenden Ausnahmen:

- Zu den signierten Attributen gehört nicht das Attribut „Content-Type“ (s. Kapitel 6.1.1), da es keinen eigenen Datentyp für Gegenzeichnungen gibt.[94]
- Bei der Gegenzeichnung werden nicht die Nutzdaten signiert (vgl. Kapitel 5.3) sondern der Inhalt des Feldes „signature“ der Datenstruktur „SignerInfo des Signierers, wodurch dessen Signatur noch einmal signiert wird. Der Input für den Signaturprozeß des Gegenzeichners ist der DER-kodierte Wert dieses Feldes (vgl. Kapitel 7).

Das Attribut „Countersignature“ kann mehrere Werte haben, es muß jedoch mindestens ein Wert vorhanden sein. Das Attribut kann auch mehrfach vorhanden sein, d.h. mehrere Instanzen des Attributs sind zulässig.

Da eine Gegenzeichnung ein Signaturblock vom Typ „SignerInfo“ ist, kann sie auch selbst wiederum Attribute vom Typ Gegenzeichnung enthalten. Es können auf diese Weise beliebige Folgen von Gegenzeichnungen konstruiert werden.

7 Signaturbildung

Die Signaturbildung erfolgt durch Anwendung eines der in Kapitel 3 beschriebenen Signaturverfahren auf die zu signierenden Daten. Es muß stets das Signaturverfahren verwendet werden, das im Feld „publicKeyInfo“ des Signaturschlüssel-Zertifikats ausgewiesen ist.

Signiert werden ausschließlich die im Feld „signedAttrs“ enthaltenen Attribute (vgl. Kapitel 5.5).

Damit die Nutzdaten in die Signatur einbezogen werden, wird in einem ersten Schritt entsprechend Kapitel 5.4 [CMS 98] ein Attribut von Typ „Message Digest“ erzeugt (vgl. Kapitel 6.1.2). Dies Attribut wird in der Regel durch Bildung des Hashwertes über den Inhalt des Feld „encapContentInfo“ der Datenstruktur „Signed Data“ generiert (s. Kapitel 5.3). Es werden dabei nur die Oktette verwendet, die den Wert ausmachen, nicht die Oktette für den Tag und die Längenangabe.[95]

Bei der Hashwertbildung ist zu beachten, daß es Beschränkungen hinsichtlich des zur verwendenden Hashalgorithmus geben kann. Es sind zwei Fälle zu unterscheiden:

- Falls im Feld „publicKeyInfo“ des Signaturschlüssel-Zertifikats die Signaturverfahren RSA mit PKCS- oder ISO9796-2-Padding ausgewiesen sind, können sowohl SHA-1 als auch RIPEMD-160 verwendet werden. Der Signierer kann frei entscheiden, welchen Hashalgorithmus er verwendet.

---

94 Da durch Gegenzeichnungen stets eine bestehende Signatur noch einmal signiert wird, ist der Datentyp implizit festgelegt.

95 Auch wenn Tag und Längenangabe nicht in den Hashwert einfließen, so sind diese Werte doch gesichert. Die Längenangabe ist durch die Eigenschaft des Hash-Algorithmus gesichert, daß es praktisch unmöglich ist, zwei verschiedene Daten zu finden, die auf den gleichen Hashwert abgebildet werden.

- Falls im Feld „publicKeyInfo“ des Signaturschlüssel-Zertifikats die Signaturverfahren RSA mit PKCS mit ISO9796-2rnd-Padding, DSA oder ECDSA ausgewiesen sind, darf der Signierer nur einen bestimmten Hashalgorithmus verwenden. Ob SHA-1 oder RIPEMD-160 verwendet werden müssen richtet sich danach, was der OID in dem Zertifikatsfeld indiziert.[96]

Im Falle externer Signaturen erfolgt die Signaturbildung über die Nutzdaten so, als wenn diese im Feld „encapContentInfo“ enthalten wären.

Der Hashwert wird anschließend als Wert des Attributs vom Typ „MessageDigest“ in das Feld „signedAttrs“ eingefügt (vgl. Kapitel 5.5).

Im nächsten Schritt werden alle Attribute der Menge der zu signierenden Attribute im Feld „signedAttrs“ der Datenstruktur „SignerInfo“ DER-kodiert. Dabei wird eine spezielle Kodierung verwendet. Statt des impliziten Tags [0] wird das explizite Tag „SET OF“ verwendet (vgl. [ITUX690 94]). Die anschließende Hashwertbildung erfolgt somit über den DER-kodierten „SET OF“-Tag statt des impliziten [0]-Tag zusammen mit der Längenangabe und dem Wert.

Bei der anschließenden Hashwertbildung ist der gleiche Hashalgorithmus zu verwenden, der bei der ersten Hashwertbildung verwendet wurde (s.o.).

Bei der anschließenden Signaturbildung ist das Signaturverfahren zu verwenden, daß im Feld „publicKeyInfo“ des Signaturschlüssel-Zertifikats ausgewiesen wird.

Der Input für die Signaturbildung (DSI) besteht aus dem Hashwert, der im Falle von RSA gepadded wird. Bei den anderen Verfahren findet kein Padding statt, d.h. der Hashwert ist der DSI. Die Details der Signaturbildung für jedes zugelassene Signaturverfahren sind in Kapitel 3.2 beschrieben.

Es darf nur das Padding-Verfahren verwendet werden, das im Feld „publicKeyInfo“ des Signaturschlüssel-Zertifikats ausgewiesen ist.

Das Ergebnis der Signaturbildung, d.h. die Signatur, wird als OCTET STRING in das Signaturfeld der Datenstruktur „SignerInfos“ eingetragen.

8 Austauschformat

Ein Austauschformat wird benötigt, um in Nachrichten eingebettete Signaturaustauschobjekte transportieren und anschließend automatisch weiterverarbeiten zu können. Falls die Nachricht per E-Mail versandt werden soll, muß sie z.B. kodiert werden, damit sie auf dem Transportweg nicht durch das Mail-Transportsystem verfälscht wird. Der Empfänger einer Nachricht benötigt für eine automatische Weiterverarbeitung z.B. Informationen über den Nachrichtentyp.

Aus SigG/SigV können keine Anforderungen hinsichtlich der Unterstützung bestimmter Austauschformate abgeleitet werden. Es besteht weder die Anforderung, daß Signaturaustauschobjekte auf definierte Art und Weise ausgetauscht werden können, noch daß eine automatische Weiterverarbeitung möglich sein muß. Deshalb sind auch Applikationen Signaturgesetzkonform, die auf die Implementierung eines Austauschformates insgesamt verzichten. In der vorliegenden Spezifikation werden daher nur Empfehlungen für die Unterstützung von Austauschformaten gegeben.

Es gibt eine Vielzahl verschiedener Standards für den Austausch von Nachrichten, die in der Praxis nebeneinander verwendet werden. Im Rahmen der vorliegenden Spezifikation erfolgt aus Gründen der Interoperabilität und der Vereinfachung der Umsetzung dieser Spezifikation jedoch eine Beschränkung auf einen der Standards.

---

96 Die OIDs sind in Kapitel 3.4.2 beschrieben.

Das in Kapitel 5 definierte Signaturaustauschformat basiert auf der Cryptographic Message Syntax (CMS), die ihrerseits zum Schutz von MIME-Nachrichten entwickelt wurde. Durch CMS geschützte MIME-Nachrichten sind S/MIME-Nachrichten, die daher im Rahmen dieser Spezifikation als Austauschformat beschrieben werden.

S/MIME-Nachrichten bestehen aus MIME-Bodies und geschützten Signaturaustauschobjekten, die in der S/MIME-Terminologie als CMS-Objekte bezeichnet werden. Im Rahmen der vorliegenden Spezifikation wird ein S/MIME-Profil gebildet, das für eine Vielzahl von Applikation geeignet erscheint, um Signaturgesetz-konforme CMS-Objekte austauschen zu können. Durch die Profilbildung wird erreicht, daß Applikationen interoperabel sind, sofern sie der vorliegenden Spezifikation entsprechen.

Die S/MIME-Spezifikation wird im Nachfolgenden nicht in Einzelheiten zitiert sondern als bekannt bzw. verfügbar vorausgesetzt. An vielen Stellen wird auf Teile der S/MIME-Spezifikation verwiesen.

8.1 S/MIME-Nachrichtentypen

Eine S/MIME-Nachricht ist eine Kombination von MIME-Bodies und geschützten CMS-Objekten. Die CMS-Objekte entsprechen im Wesentlichen den in Kapitel 5 definierten allgemeinen Signaturaustauschobjekten. Es ist jedoch zu beachten, das die Nutzdaten bei CMS-Objekten stets in Form eines kanonischen MIME-Objekts vorliegen müssen.[97] Es muß somit eine Kanonisierung erfolgen (vgl. Kapitel 8.2.1).

Das kanonische MIME-Objekt wird dann wie in Kapitel 5 beschrieben in ein CMS-Objekt eingebettet, falls keine externe Signatur gebildet werden soll. Das CMS-Objekt wird anschließend in eine S/MIME-Nachricht eingefügt.

Im einfachsten Fall besteht eine S/MIME-Nachricht genau aus einem einfachen MIME-Objekt, das in ein CMS-Objekt eingebettet ist und dem MIME-Felder zum Zwecke der Identifizierung des Nachrichtentyps und des Transportes hinzugefügt werden. Das MIME-Objektes kann jedoch auch aus mehreren Teilen zusammengesetzt sein (multipart). Auch können S/MIME-Nachrichten ihrerseits als Nutzdaten in ein weiteres CMS-Objekt eingebettet werden. Es sind somit beliebige Verschachtelungen möglich.

Jede MIME-Nachricht wird durch die Angabe eines Nachrichtentyps in einem MIME-Kopffeld identifiziert. Der Typ jeder MIME-Nachricht wird durch ihren „Content-Type“ und ggf. weitere Parameter festgelegt. Der Content-Type besteht aus einem Medientyp (Media Type) und einem Subtyp, der das spezifische Format bestimmt [RFC 2045 96].

Es wurden bisher folgende fünf Medientypen spezifiziert [RFC 2046 96]:

- text (für textuelle Nachrichten)
- image (für grahische Nachrichten)
- audio (für Audio-Daten)
- video (für Video-Daten)
- application (für alle anderen Arten von Daten, z.B. uninterpretierte Binärdaten)

Daneben sieht MIME zwei Medientypen für zusammengesetzte Nachrichten vor:

- multipart (für mehrere verschiedene Datentypen)
- message (für gekapselte Nachrichten)

---

97 Bei der Definition des Signaturaustauschformates wurden kein Anforderungen an das Format der Nutzdaten gestellt.

Für jeden Medientyp gibt es eine Vielzahl von Subtypen. Durch die Angabe von „text/plain“ als Content-Type wird z.B. der Typ von Textnachrichten beschrieben, der durch [RFC 822 82] definiert ist.

Die S/MIME-Spezifikation beschreibt vier Nachrichtentypen für verschlüsselte und signierte Nachrichten. Der Content-Type „application/pkcs7-mime“ kann sowohl für verschlüsselte als auch für signierte Nachrichten verwendet werden. Der Typ kann über den Parameter „smime-typ“ genauer spezifiziert werden. Daneben besteht die Möglichkeit, die Content-Types „multipart/signed“ und „multipart/encrypted“ zu verwenden.

Der Content-Type „application/pkcs7-mime“ [98] sollte stets zusammen mit dem Parameter „smime-typ“ verwendet werden. Dadurch wird es dem Empfänger erspart, eine ASN.1-Dekodierung vorzunehmen, um festzustellen, ob es sich um verschlüsselte oder signierte Daten handelt.[99] Für den Parameter sind folgende Werte definiert:

- „smime-type=enveloped-data“ für verschlüsselte Daten
- „smime-type=signed-data“ für signierte Daten

Nachrichten vom Typ „multipart/signed“ (oder „multipart/encrypted“) bestehen im Gegensatz zu den o.g. Nachrichtentypen aus zwei Teilen (body parts).[100] Der erste Teil enthält die Daten, die digital signiert (bzw. verschlüsselt) sind. Der zweite Teil enthält die Kontrollinformationen, die zur Verifikation der digitalen Signatur (bzw. zur Entschlüsselung) verwendet werden. Der Vorteil dieser Aufteilung ist, daß die Teile unterschiedlich kodiert werden können.

Diese Aufteilung ist dann sinnvoll, wenn Empfängern die Möglichkeit eingeräumt werden soll, S/MIME-Nachrichten auch ohne S/MIME-Software mit einem Standard-Viewer betrachten zu können. Bei signierten Nachrichten kann es z.B. sinnvoll sein, eine Nachricht mit lesbarem Klartext zu übertragen. Entsprechend [SMIME MSG 98] sollte der Typ „multipart/signed“ aus diesem Grunde generell bevorzugt werden.

8.1.1 application/pkcs7-mime; smime-type=signed-data

Mit dem Nachrichtentyp „application/pkcs7-mime; smime-type=signed-data“ können beliebige Daten so bearbeitet werden, daß sie authentisch und integer übertragen werden können. Das Schutzobjekt kann ein beliebiger MIME-Objekt sein.

Der Nachrichtentyp kann für jede Art zu schützender MIME-Nachricht verwendet werden. Durch die explizite Angabe des Typs der geschützten Daten erlaubt MIME, daß Komponenten die Daten dem Nutzer in aufbereiteter Form präsentieren oder auf andere geeignete Weise damit umgehen können.

Folgende Transformationsschritte, die in Kapitel 8.2 genauer beschrieben werden, können bei der Erstellung einer signierten Nachricht durchzuführen sein:

- Kanonisieren
- Generieren eines CMS-Objektes
- Kodieren
- Zusammensetzen

---

98 Dieser Typ wird in [SMIME-MSG 98] definiert.

99 Auch wenn verschlüsselte Nachrichten nicht Gegenstand der vorliegenden Spezifikation sind, so ist doch nicht ausgeschlossen, daß eine Signaturgesetz-konforme Applikation auch verschlüsselte Nachrichten generieren und verarbeiten kann.

100 Diese Typen werden in [RFC 1847 95] definiert.

Durch das Kanonisieren werden die Nutzdaten vor der Signaturbildung in eine Form transformiert, die der Empfänger eindeutig interpretieren kann, falls erforderlich.

Das Generieren eines CMS-Objektes erfolgt wie bereits in Kapitel 5 beschrieben.

Da das CMS-Objekt in binärer Form vorliegt, muß eine Kodierung erfolgen, falls nicht für die Übertragung ein 8-Bit-transparentes Übertragungsmedium verwendet wird.

Falls eine Kodierung erfolgt, so umfaßt sie die gesamte Nachricht einschließlich der Kopfzeilen. Falls der Nachrichtentext lesbar bleiben soll, ist in diesem Fall der Nachrichtentyp „multipart/signed“ zu verwenden.

8.1.2 multipart/signed

Mit dem Nachrichtentyp „multipart/signed“ können beliebige Daten so bearbeitet werden, daß sie authentisch und integer übertragen werden können. Das Schutzobjekt kann eine beliebige MIME-Nachricht sein.

Nachrichten vom Typ „multipart“ bestehen aus zwei Teilen, die hinsichtlich der Kodierung verschieden behandelt werden können. Dies ist der wesentliche Unterschied zu Nachrichten vom Typ „application/pkcs7-mime; smime-type=signed-data“. Die Ausführungen zu diesem Nachrichtentyp gelten mit dieser Ausnahme entsprechend.

8.2 Transformation von Nachrichten

Bei der Transformation von Nachrichten wird das zu schützenden MIME-Objekt gesichert und in eine S/MIME-Nachricht eingebettet. Das zu schützende MIME-Objekt ist im einfachsten Fall eine MIME-Nachricht, bestehend aus einem MIME-Header und einem MIME-Body.[101] Das zu schützende Objekt kann jedoch auch aus mehreren Teilen bestehen, die ihrerseits wiederum verschachtelt sein können. Transformiert wird jeweils das „innerste Objekt“ einer möglicherweise umfangreichen MIME-Nachricht.

8.2.1 Vorbereitung des MIME-Objekts

Jedes MIME-Objekt muß zunächst für die Signaturbildung vorbereitet werden. Dazu kann eine Kanonisierung und eine Kodierung erforderlich sein.

Die Art der Kanonisierung ist vom Medien-Typ abhängig. Auf eine Beschreibung der Kanonisierung wird hier verzichtet. Auf den Standard für den jeweiligen Medien-Typ wird verwiesen.

Eine Kodierung des MIME-Objektes ist nur im Falle des Nachrichtentyps „multipart/signed“ erforderlich. Der MIME-Body einer Nachricht vom Typ „multipart/signed“ besteht aus genau zwei Teilen (body parts). Der erste Teil enthält die Daten, über die die digitale Signatur gebildet wird. Der zweite Teil enthält die Kontrollinformationen zur Verifikation der digitalen Signatur. Der erste Teil kann jeden beliebigen Content-Type enthalten.

Ob eine Kodierung erforderlich ist, richtet sich nach der Form, in der die Daten vorliegen und den Eigenschaften des Transportmediums. Falls die Daten z.B. bereits als Textnachrichten (Content-Typ „text/plain“) vorliegen, ist eine Kodierung nicht erforderlich. Eine Kodierung ist auch nicht erforderlich, wenn ein 8-Bit-transparentes Übertragungsmedium verwendet werden soll.

---

101 Im Falle von Internet-Mail umfaßt die Nachricht nicht den RFC-822-Header.

Für MIME-Nachrichten sind drei Kodierungsvarianten definiert [RFC 2045 96]. Die gewählte Kodierungsvariante wird im Feld „encoding“ des MIME-Headers angegeben. Die Kodierungsvarianten sind:

- Identität
- „quoted-printable“
- „base64“

Identität bedeutet, daß keine Kodierung erfolgt. Die Kodierungsvariante „quoted-printable“ ist für Daten gedacht, die im wesentlichem dem Zeichensatz US-ASCII entsprechen. In der Regel wird jedoch die Kodierungsvariante „base64“ verwendet. Für eine Darstellung der Kodierungsvarianten wird auf [RFC 2045 96] verwiesen.

8.2.2 Generieren eines CMS-Objektes

In diesem Transformationsschritt wird wie bereits beschrieben das CMS-Objekt unter Verwendung des vorbereiteten MIME-Objekts generiert.

8.2.3 Kodieren

Nachdem durch Signatur oder Verschlüsselung ein CMS-Objekt gebildet worden ist, kann zunächst eine Kodierung des CMS-Objektes, das in binärer Form vorliegt, erforderlich sein.

Die Kodierung von Nachrichten des Typs „multipart/signed“ wurde bereits in Kapitel 8.2.1, „Vorbereitung des MIME-Objekts“, behandelt. Die dort angegebenen Regeln gelten auch für die Kodierung des CMS-Objekts.

8.2.4 Zusammensetzen

Das CMS-Objekt muß abschließend in eine MIME-Nachricht eingebettet werden. Im einfachsten Fall besteht die resultierende MIME-Nachricht aus einem MIME-Header und einem MIME-Body, die durch eine Leerzeile voneinander getrennt sind, z.B.:

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-data
     Content-Transfer-Encoding: base64
<Leerzeile>
rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V
```

Der Body einer S/MIME-Nachricht enthält außer bei externen Signaturen das gesamte CMS-Objekt, das neben den Nutzdaten alle weiteren Informationen enthält, die der Empfänger benötigt.

Die Nachricht enthält keine Abgrenzungszeilen. Abgrenzungszeilen sind nur dann erforderlich, wenn mehrere Teile zu einer Nachricht zusammengefaßt werden, wie das folgende Beispiel zeigt:

```
Content-Type: multipart/signed;
     protocol="application/pkcs7-signature";
     micalg="micalg"; boundary=boundary42

--boundary42
Content-Type: text/plain
```

```
Die ist eine lesbare Textnachricht.

--boundary42
Content-Type: application/pkcs7-signature
Content-Transfer-Encoding: base64

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42--
```

Durch Angabe einer Wertes für den Parameter „boundary“ kann eine beliebige Abgrenzungszeile gebildet werden. Anfang und Ende können dadurch unterschieden werden, daß bei einer Ende-Zeile zwei Bindestriche (dezimaler Wert 45) angehängt werden.

Parameterwerte für Abgrenzungszeilen sind aufgrund der Möglichkeit der Bildung von Multipart-Nachrichten und der Verschachtelung erforderlich. Die gewählten Parameterwerte müssen sich voneinander unterscheiden.

Bei einer Nachricht vom Typ „multipart/signed“ muß der Parameter „protocol“ stets so wie im Beispiel (mit den Anführungsstrichen) angeben werden.

Bei einer Nachricht vom Typ „multipart/signed“ handelt es sich um einen Fall einer externen Signatur, da die Nutzdaten nicht im CMS-Objekt enthalten sind, sondern in einem eigenen Body-Part.

Durch die Angabe des Parameters „micalg“ wird die Prüfung der Signatur in einem Durchgang ermöglicht. Mit der Hashwertbildung über die signierte Nachricht kann sofort begonnen werden, ohne daß zunächst der zweite Nachrichtenteil untersucht werden muß, um den OID der verwendeten Hashfunktion zu bestimmen. Die Hashfunktion wird über den Parameter „micalg“ übergeben.[102] Von dieser Option sollte stets Gebrauch gemacht werden.

9 Literatur

[AgMuVa 93]	G. Agnew, R. Mullin and S. Vanstone: "An implementation of elliptic curve cryptosystems over F2155"; IEEE Journal on Selected Areas in Communications, 11 (1993), 804-813
[ANSI X.31 95]	American National Standard for Financial Services – Public key cryptography using RSA for the financial services industrie – Part 1: The RSA signature algorithm; draft; 1995
[ANSI X9.57 97]	American National Standard for Financial Services – Public key cryptography for the financial services industrie: Certificate Management, February 1997
[ANSI X9.62 99]	American National Standard for Financial Services – Public key cryptography for the financial services industrie : The Elliptic Curve Digital Signature Algorithm (ECDSA); January 1999

[102] Selbstverständlich muß nach der Hashwertbildung geprüft werden, ob der über den Parameter „micalg“ angegebene Algorithmus tatsächlich mit dem Algorithmus übereinstimmt, der durch die Signatur geschützt im zweiten Nachrichtenteil enthalten ist.

[BAZ140298]	Bundesanzeiger vom 14. Februar 1998, Nr.31, 1787f.: Bekanntmachung zur digitalen Signatur nach Signaturgesetz und Signaturverordnung
[BSI 97]	Maßnahmenkatalog für digitale Signaturen - auf Grundlage von SigG und SigV, 1997
[BSI-DIR 99]	BSI: Spezifikation zur Entwicklung interoperabler Verfahren und Komponenten nach SigG/SigV; Verzeichnisdienst (in Bearbeitung); Stand 31.01.1999
[BSI-GM 99]	BSI: Spezifikation zur Entwicklung interoperabler Verfahren und Komponenten nach SigG/SigV; Gültigkeitsmodell (in Bearbeitung)
[BSI-TSP 99]	BSI: Spezifikation zur Entwicklung interoperabler Verfahren und Komponenten nach SigG/SigV; Zeitstempel (in Bearbeitung); Stand 31. März 1999
[BSI-ZERT 99]	BSI: Spezifikation zur Entwicklung interoperabler Verfahren und Komponenten nach SigG/SigV; Zertifikate (in Bearbeitung); Stand 31.01.1999
[Chok]	S. Chokhani: A Security Flaw in the X.509 Standard; CygnaCom Solutions, Inc.; http://www.cygnacom.com
[CMS98-v10]	S/MIME Working Group: Cryptographic Message Syntax; (work in progress) Stand: Dezember 1998
[CMS 98]	S/MIME Working Group: Cryptographic Message Syntax; (work in progress) Stand: Oktober 1998
[DINChip 98]	DIN Spezifikation der Schnittstelle zu Chipkarten mit Digitaler Signatur-Anwendung/Funktion nach SigG und SigV, (in Bearbeitung) Version 0.99, Stand: September 1998
[DBP 96]	H. Dobbertin, A. Bosselaers, B. Preneel: „RIPEMD-160, a strengthened version of RIPEMD.''; Fast Software Encryption Workshop, D. Gollmann, Ed., Springer-Verlag, 1996, LNCS 1039, pp. 71-82.
[DSS 94]	NIST: FIPS Publication 186: Digital Signature Standard (DSS); Mai 1994
[Elg 85]	T. El'gamal: A public key cryptosystem and a signature scheme based on discrete logarithms; Crypto '84, Springer-Verlag, 1985, LNCS 196, pp. 10 - 18
[ESS 99]	S/MIME Working Group: Enhanced Security Services for S/MIME; (work in progress) Stand: Februar 1999
[FIPS180-1 95]	Federal Information Processing Standards (FIPS PUB) 180-1: Secure Hash Standard; April 1995
[FIPS186-1 98]	Federal Information Processing Standards (FIPS PUB) 186-1: Digital Signature Standard; December 1998
[Fox 98]	D. Fox: Zu einem prinzipiellen Problem digitaler Signaturen; Datenschutz und Datensicherheit 7/1998
[IEEE P1363 98]	IEEE P1363 / D8 (Draft Version 8) Standard Specifications for Public Key Cryptography,;Oktober 1998

[ISO 9796-2 97]	ISO/IEC 9796-2: IT - Security techniques - Digital signature scheme giving message recovery - Part 2: Mechanisms using a hash-function; 1997
[ISO10118-3 98]	International Organization for Standardization: IT-Security techniques Hash-Functions Part 3: Dedicated Hash-Functions; Standard ISO/IEC 10118, 1998
[ISO14888-1 98]	International Organization for Standardization: Information technology - Security techniques - Digital signatures with appendix - Part 1: General, ISO/IEC FDIS 14888-1, 15 Juni 1998
[ISO 14888-3 98]	International Organization for Standardization: Information technology - Security techniques - Digital Signatures with Appendix - Part 3: Certificate-based mechanisms; Standard ISO/IEC 14888-3 FDIS, 1998
[ISO15946-2 99]	International Organization for Standardization: Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 2: Digital signatures, ISO/IEC WD 15946-2, 10. Februar 1999
[ITU-T X.509 97]	ITU-T X.509: Information Technology - Open Systems Interconnection - The Directory: Authentication Framework; 1997
[ITUX680 97]	ITU-T X.680: Information technology – Abstract syntax notation one (ASN.1); 1997
[ITUX690 94]	ITU-T X.690: Information Technology - ASN.1 Encoding Rules - Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER); 1994
[MeOV 93]	Menezes, Alfred J.; Okamoto, Tatsuaki; Vanstone, Scott A.: Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. IEEE Transactions on Information Theory, Vol. 39, No. 5, Sept. 1993, S. 1639-1646
[MISPC1 97]	NIST's Cooperative Research and Development Agreements for Public Key Infrastructure development : MISPC Minimum Interoperability Specification for PKI Components; Version 1, September 1997
[Odl 95]	A. M. Odlyzko: The Future of Integer Factorization, Cryptobyte Summer 95; 1995
[PKCS1 98]	RSA Laboratories: PKCS #1 v.2.0: RSA Cryptography Standard; RSA Laboratories; Oktober 1998
[RegTP12A2]	Regulierungsbehörde für Telekommunikation und Post: Maßnahmenkatalog nach § 12 (2) SigV; 1998
[RegTP16A6]	Regulierungsbehörde für Telekommunikation und Post: Maßnahmenkatalog nach § 16 (6) SigV; 1998
[RFC 822 82]	Crocker, D. H.: Standard for the Format of ARPA Internet Text Messages; August 1982
[RFC 1847 95]	Galvin, G; Murphy, S.; Crocker, S.; Freed, N.: Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted; Oktober 1995
[RFC 2045 96]	Freed, N; Borenstein, N: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Mail Bodies, RFC2045; November 1996

[RFC 2046 96]	Freed, N; Borenstein, N: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, RFC2046; November 1996
[RFC 2183 97]	Troost, R.; Dorner, S.; Moore, K.: Communicating Presentation Information in Internet Messages: The Content-Dispositon Header Field; August 1997
[RIPE 92]	Research and Development in Advanced Communication Technologies in Europe, RIPE Integrity Primitives: Final Report of RACE Integrity Primitives Evaluation (R1040), RACE; June 1992
[RIPEMD-160 96]	H. Dobbertin, A. Bosselaers und B. Preneel: „RIPEMD-160: A strengthened version of RIPEMD“; Fast Software Encryption - Cambridge Workshop 1996, LNCS, Band 1039, S. 71 - 82, Springer-Verlag; April 1996
[RiShAd 78]	R. Rivest, A. Shamir, L. Adleman: A method for obtaining digital signatures and public key cryptosystems, Communications of the ACM, vol. 21 no. 2, 1978
[SHS 95]	NIST: FIPS Publication 180-1: Secure Hash Standard (SHS-1); Mai 1995
[SigG]	Gesetz zur Regulierung der Rahmenbedingungen für Informations- und Kommunikationsdienste (IuKDG), Artikel 3 – Gesetz zur digitalen Signatur (Signaturgesetz – SigG); BGBI. I S. 1870, 1997
[SigV]	Verordnung zur digitalen Signatur (Signaturverordnung – SigV); 1997
[SMIME-MSG 98]	Ramsdell, B.: S/MIME Version 3 Message Specification; August 1998
[TTT-AF 99]	TeleTrusT - MailTrusT Version2: Austauschformat; März 1999

Anhang A: Datentypen

In [CMS 98] wird nur ein Datentyp mit folgendem OID identifiziert:

```
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs7(7) id-
data(1) }
```

Dieser Datentyp, der abgekürzt mit „id-data“ bezeichnet wird, wird für allgemeine binäre Daten verwendet, deren Datentyp unbekannt bzw. nicht näher spezifiziert ist. Die Interpretation dieser Daten wird der Applikation überlassen. Der Empfänger von Daten dieses Typs benötigt implizite Informationen, um die Daten Auswerten zu können.

Ein weiterer OID für den Datentyp „Zeitstempel“ wird in Kapitel 6.2 [BSI-TSP 99] definiert. Ein entsprechender OID für Zeitdatenstempel wird in Kapitel 6.4 [BSI-TSP 99] definiert.

Die nachfolgende Tabelle enthält die im Rahmen dieser Spezifikation definierten OIDs für allgemein verwendete Datentypen. Diese OID sind noch zu registrieren.

Die Tabelle erhebt keinen Anspruch auf Vollständigkeit. Die Registrierung weiterer OIDs ist jederzeit möglich.

Alle für SigI definierten OIDs gehören zum Teilbaum, der durch folgenden OID definiert wird:

```
Id-sigi                    OBJECT IDENTIFIER ::= { 1 3 36 8 }
```

Alle OIDs für die Datentypen gehören zu folgendem Teilbaum:

```
Id-sigi-sigtype              OBJECT IDENTIFIER ::= { 1 3 36 8 X[103]}
(Kurzbezeichner „type“)
```

Die folgende Tabelle enthält die definierten OIDs:

OID	Datentyp
{ type Adobe-ILL (1) }	Adobe Illustrator
{ type AmiPro (2) }	Ami Professional
{ type AutoCAD (3) }	AutoCAD DXF
{ type Binary (4) }	Allgemeine Binärdaten
{ type BMP (5) }	Windows/OS-2 Bitmap
{ type CGM (6) }	Computer Graphics Metafile
{ type CorelCRT (7) }	CorelCHART
{ type CorelDRW (8) }	CorelDRAW
{ type CorelEXC (9) }	Corel Presentation Exchange
{ type CorelPHT (10) }	CorelPHOTO-PAINT
{ type Draw (11) }	Micrographx Draw
{ type DVI (12) }	Device Independent Format, DVI-Datei
{ type EPS (13) }	(Placable) Encapsulated Postscript
{ type Excel (14) }	Excel-Tabelle

[103] Der Wert für „sigtype“ ist noch festzulegen.

OID	Datentyp
{ type GEM (15) }	GEM-Dateien
{ type GIF (16) }	Compuserve Bitmaps, „GIF„-Datei
{ type HPGL (17) }	HPGL Plotter Datei
{ type JPEG (18) }	JPEG Bitmap
{ type Kodak (19) }	Kodak Photo-CD
{ type LaTeX (20) }	LaTeX-Datei
{ type Lotus (21) }	Lotus 123 1A
{ type Lotus-PIC (22) }	Lotus PIC
{ type Mac-PICT (23) }	Macintosh PICT
{ type Mac-Word (24) }	Microsoft Word für Macintosh
{ type MS-WfD (25) }	Microsoft Word für DOS
{ type MS-Word (26) } { type MS-Word-2 (27) } { type MS-Word-6 (28) } { type MS-Word-8 (29) }	Microsoft Word für Windows Microsoft Word für Windows, Version 2.0 Microsoft Word, Version 6,7,95 Microsoft Word, Version 8,97
{ type PDF (30) }	Adobe Portable Data Format
{ type PIF (31) }	IBM Program Information File
{ type Postscript (32) }	(Interpreted) Postscript
{ type RTF (33) }	Microsoft Rich Text Format
{ type SCITEX (34) }	SCITEX
{ type TAR (35) }	UNIX Tape ARchive
{ type Targa (36) }	TARGA Bitmap
{ type TeX (37) }	Plain-TeX Datei
{ type Text (38) }	ASCII-Text-Datei
{ type TIFF (39) }	TIFF-Bitmap
{ type TIFF-FC (40) }	TIFF Four-Color Datei
{ type UID (41) }	User Interface Database, UID-Datei (OSF/Motif)
{ type UUEncode (42) }	UUEncode-te Datei
{ type WMF (43) }	Windows Metafile
{ type WordPerfect (44) }	WordPerfect
{ type WP-Grph (45) }	WordPerfect Graphic

Tabelle 3: OIDs für Datentypen