MOA SP-SS

Spezifikation
Module für Online Anwendungen – SP und SS

Dokumentinformation

Bezeichnung	Spezifikation Module für Online Anwendungen – SP und SS
Kurzbezeichnung	MOA SP-SS
Version	2.0.0
Datum	15.05.2013
Dokumentenklasse	Konvention
Dokumentenstadium	Öffentlicher Entwurf
Kurzbeschreibung	Dieses Dokument spezifiziert zwei essentielle Servermodule für die weitere Umsetzung der E-Government-Strategien. Im Basismodul SP werden die gesamten Funktionalitäten zur Signaturprüfung gekapselt, so dass Online-Applikationen komfortabel auf diese Basis-Funktionalitäten zugreifen können. Es werden dabei sowohl Security-Layer konforme Signaturen (XAdES/CAdES) als auch XMLDSig und CMS Signaturen unterstützt. Das Basismodul SS kapselt analog dazu die gesamte Funktionalität zur serverseitigen Signaturerstellung. Mit MOA SS können CMS und XML-Signaturen erzeugt werden, die bei Bedarf auch den Anforderungen des Security-Layers (CAdES/XAdES) entsprechen. Mit dem vorliegenden Dokument wird eine weitere Vereinheitlichung des E-Governments sowie eine Vereinfachung der Umsetzung von modernen Online-Anwendungen angestrebt.
Autoren	Stabsstelle IKT-Strategie des Bundes im Bundeskanzleramt in Kooperation mit dem Bundesministerium für Finanzen Email: moa@cio.gv.at Ab Version 1.5.2: EGIZ – E-Government Innovationszentrum Email: post@egiz.gv.at

Inhaltsverzeichnis

Dokumenten Information

Versionen

Version	Datum	Beschreibung	Autor
2.0.0	15.05.2013	• Erweiterung der Schnittstelle zur Erzeugung von CMS/CAdES Signaturen • Update der Anforderungen an das Signaturformat (SignatureMethod und DigestMethod)	EGIZ
1.3.0	24.08.2005	• Änderungen beim Retournieren der Referenz-Eingangsdaten bzw. Hash-Eingangsdaten im `VerifyXMLSignatureResponse`; Anpassung der Beschreibungen in den Abschnitten 5.1.2.1.2, 5.1.2.1.3, 5.1.3.1.2, 5.1.3.1.3 geändert.	CIO
1.2.2	07.05.2005	• Änderung der Semantik des Elements `SignatureManifestCheck` im `VerifyXMLSignatureRequest`; Anpassung der Beschreibungen in den Abschnitten 5.1.2.1.2, 5.1.3.1.3 und5.1.3.1.4.	CIO
1.2.1	20.01.2005	• Korrektur der Algorithmus-URI für ECDSA-Signaturen (Abschnitt 5.1.6)	CIO
1.2	30.06.2004	• Erweiterung der Möglichkeiten Daten in den Befehlen zu spezifizieren (LocRefContent).	CIO
1.1	30.06.2003	Abnahmeversion	BMF/CIO
1.0	27.08.2004	Erstversion	BMF/CIO

1 Einleitung

Die vorliegende Spezifikation von MOA umfasst die Module *Signaturprüfung* und *Server-Signatur*.

Zielsetzung dieses Dokuments ist die genannten Module so zu spezifizieren, dass eine Implementierung in Phase 2 durchgeführt werden kann. Stellt sich in Phase 2 heraus, dass Ergänzungen sinnvoll sind, so können diese während dieser Phase 2 durchgeführt werden.

Diese Spezifikation baut so weit wie möglich auf der Spezifikation des Security-Layers [SecLayer1.2x] auf. Abweichungen zum Security-Layer werden dort vorgeschlagen wo dies sinnvoll erscheint.

Gewünschte Funktionalität, für die derzeit eine Spezifikation nicht sinnvoll erscheint wird in *Abschnitt 10 Anhang – Nicht-Spezifizierte Funktionalität* mit der entsprechenden Begründung aufgelistet.

2 Anwendungsfälle

Die Dokumente [UC CIO] und [UC BRZ] beschreiben die Anwendungsfälle aus Sicht der Online-Applikationen, die MOA abdecken muss.

Die Anwendungsfälle, die MOA abdecken muss sind in der folgenden Liste zusammengefasst:

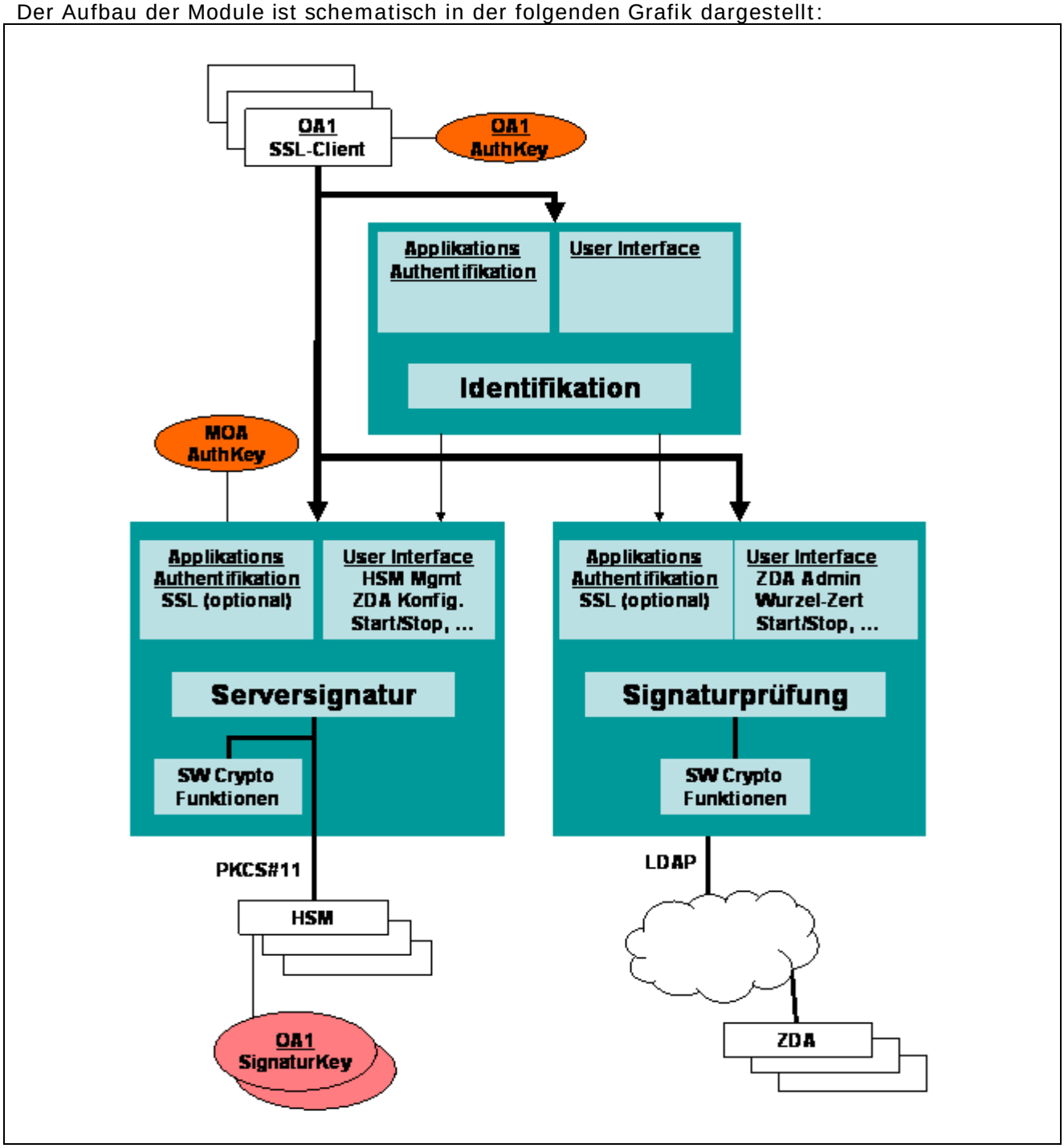
- Überprüfung einer einfachen und sicheren
  - XMLDSig Signatur, die Security-Layer konform ist (*Abschnitt 5.1 Signaturprüfung nach XMLDSig*),
  - XMLDSig Signatur (*Abschnitt 5.1 Signaturprüfung nach XMLDSig*),
  - CMS Signatur, die Security-Layer konform ist (*Abschnitt 5.2 Signaturprüfung nach CMS*) und
  - CMS Signatur (*Abschnitt 5.2 Signaturprüfung nach CMS*).

- Erstellung einer einfachen
  - XMLDSig Signatur, die Security-Layer konform ist (*Abschnitt 6 Modul Server-Signatur*).
  - CMS Signatur, die Security-Layer konform ist (*Abschnitt 6 Modul Server-Signatur*)

- Konfiguration der MOA

3 Gesamt-Überblick

Der Aufbau der Module ist schematisch in der folgenden Grafik dargestellt:



Legende:	OA	Online Applikation
	AuthKey	TLS Authentifikations-Key
	MOA	Module für Online Applikation
	HSM	Hardware Security Modul
	OCSP	Online Certificate Status Protocol
	ZDA	Zertifizierungsdiensteanbieter

Alle Module müssen sowohl gemeinsam auf einem Rechner ablauffähig sein, als auch getrennt.

4 Allgemeine Anforderungen

4.1 Unterstützte Plattformen

Es muss Java Runtime Environment ab Version 1.5 unterstützt werden.

4.2 Authentisierung

Die aufrufende Applikation kann sich mittels TLS [TLS] Client Authentisierung MOA gegenüber authentisieren. MOA kann sich mittels TLS gegenüber der aufrufenden Applikation authentisieren.

Für jede Richtung muss die TLS Authentisierung konfigurierbar sein, d.h. getrennt ein- und ausschaltbar sein.

4.3 Skalierbarkeit und Verfügbarkeit

MOA muss skalierbar und auf einen 7 * 24h Betrieb ausgelegt sein.

Modul Server-Signatur

Werde HSMs eingesetzt so gelten folgende Formulierungen:

- Die privaten Schlüssel im HSM müssen nicht doppelt verfügbar sein, da jede Applikation für die Signaturerstellung Schlüssel aus einer Schlüsselgruppe (s. *Abschnitt 6.1.2.1.3 KeyIdentifier*) verwenden kann.

- Es muss möglich sein, sowohl mehrere HSMs anschließen zu können, als auch mehrere Instanzen von MOA auf mehreren Rechnern gleichzeitig ausführen zu können die wiederum auf diese HSMs zugreifen können müssen.

- **Design-Vorschlag**
  Es wird empfohlen diese Anforderungen über die Realisierung mittels eines Clusters umzusetzen mit der Möglichkeit HSMs gemeinsam zu nutzen.

4.4 Logging

Logging soll zur Problem- und Fehlersuche im Betrieb der Module dienen.

Für jede Anfrage werden zumindest folgende Informationen mitgeloggt:

- TLS Session Identifikation (falls Verwendung von TLS konfiguriert wurde)
- Zeitpunkt des Eintreffens der Anfrage
- Zeitpunkt des Beenden der Anfrage
- Name des Requests
- Parameter des Requests
- Return-Code des Requests
- Im Falle eines Fehlers:
  - Wichtigkeit des Fehlers
  - SOAP Fault Code Mechanismus
  - Verbale Beschreibung des Fehlers
- Transaktions Identifikation – eine Nummer, die für alle Log-Einträge der selben Anfrage gleich ist

4.5 Namespace

Alle in dieser Spezifikation definierten XML-Elemente sind dem Namespace http://reference.e-government.gv.at/namespace/moa/20020822# zugeordnet.

5 Modul Signaturprüfung

Dieses Modul dient zum Überprüfen von XMLDSig und CMS bzw. XAdES und CAdES Signaturen. Diese Signaturen können einfache oder sichere Signaturen sein. Das Modul Signaturprüfmodul muss geeignet sein, auch eine sichere Signaturprüfung durchzuführen.

Es sind die geforderten Standards zu unterstützen, zusätzlich muss eine Prüfung von Signaturen möglich sein, die Zertifikate eines in Österreich akkreditierten ZDAs verwenden.

5.1 Signaturprüfung nach XMLDSig

5.1.1 Zielsetzung

Es müssen sowohl XAdES Signaturen, die der Security-Layer Spezifikation [SecLayer1.2x] entsprechen, als auch XMLDSig Signaturen, die der W3C Recommendation XMLDSig [XMLDSig] entsprechen, überprüft werden können. Einschränkungen an das Signaturformat in letzterem Fall werden in *Abschnitt 5.1.6 Anforderungen an das Signaturformat* spezifiziert.

5.1.2 Anfrage

Die Anfrage orientiert sich an dem Message-Format für `<VerifyXMLSignatureRequest>` des Security-Layer.

5.1.2.1 Abweichung vom Security-Layer

5.1.2.1.1 Angabe von Daten

Für die Elemente `VerifySignatureInfo/VerifySignatureEnvironment` und `SupplementProfile/Content` steht neben den auch von der Security-Layer Spezifikation bekannen Möglichkeiten `XMLContent` und `Base64Content` die dritte Möglichkeit des Elements `LocRefContent` zur Verfügung. Als Inhalt von `LocRefContent` muss eine URL angegeben werden, die von MOA SP aufgelöst wird, um an die zu verwendenden Daten zu kommen.

5.1.2.1.2 Angabe ob auf Security-Layer-Konformität überprüft werden muss

Wird das Element `VerifyXMLSignatureRequest/SignatureManifestCheckParams` nicht angegeben, so führt MOA SP eine Prüfung eines ggf. vorhandenen Signaturmanifests in der Form durch, wie sie in Abschnitt 3.2.2, Unterabschnitt *Prüfung des Signaturmanifests* von [SecLayer1.2] gefordert ist.

Wird das Element `VerifyXMLSignatureRequest/SignatureManifestCheckParams` angegeben, liefert MOA SP in der Befehlsantwort parametrisierbar zusätzliche Informationen zurück, bzw. führt MOA SP zusätzlich zur oben beschriebenen Prüfung des ggf. vorhandenen Signaturmanifests noch weitere Prüfungen durch:

- Wird das Attribut `SignatureManifestCheckParams/@ReturnReferenceInputData` nicht angegeben oder auf den Wert `true` gesetzt, liefert MOA SP in der Befehlsantwort für die geprüfte Signatur sowohl für jede `dsig:Reference` aus `dsig:SignedInfo`, als auch für jede `dsig:Reference` aus einem `dsig:Manifest`, auf das mittels des Attributs `Type="http://www.w3.org/2000/09/xmldsig#Manifest"` in einer `dsig:Reference` aus `dsig:SignedInfo` verwiesen wird, die Referenz-Eingangsdaten zurück. Die Referenz-Eingangsdaten sind jene Daten, die als Input für die Berechnung der ersten Transformation (bzw. falls keine Transformationen spezifiziert sind, als Input für die Berechnung des Hashwertes) verwendet werden. Wird das Attribut auf den Wert `false`

gesetzt, werden diese Informationen von MOA SP in der Befehlsantwort nicht geliefert. Für die Spezifikation der Referenz-Eingangsdaten in der Befehlsantwort siehe Abschnitt 5.1.3.1.3.

- MOA SP prüft für jedes dsig:Reference Element in dsig:SignedInfo der zu prüfenden Signatur, ob die im dsig:Reference Element spezifizierte Transformationsfolge einer Folge aus einer Menge von erlaubten Transformationsfolgen entspricht.
    - Ausgenommen von dieser Prüfung wird ein dsig:Reference Element dann, wenn sein Type Attribut einen der folgenden Werte hat:
        - http://www.buergerkarte.at/specifications/Security-Layer/20020225#SignatureManifest
        - http://www.w3.org/2000/09/xmldsig#Manifest
        - http://uri.etsi.org/01903/v1.1.1#SignedProperties
    - Die Menge von erlaubten Transformationsfolgen muss je dsig:Reference in je einem SignatureManifestCheckParams/ReferenceInfo Element angegeben werden. Die Reihenfolge der ReferenceInfo Elemente muss der Reihenfolge der dsig:Reference Elemente in dsig:SignedInfo der zu prüfenden Signatur entsprechen. Für die von der Prüfung ausgenommenen dsig:Reference Elemente darf auch keine Menge von erlaubten Transformationsfolgen angegeben werden.
        - ReferenceInfo enthält eine oder mehrere erlaubte Transformationsfolgen. Für jede erlaubte Folge muss entweder ein Kindelement VerifyTransformsInfoProfile oder VerifyTransformsInfoProfileID angegeben werden. Letztere Variante bezieht sich auf ein VerifyTransformsInfoProfile, das in der Konfiguration von MOA SP hinterlegt ist.
            - VerifyTransformsInfoProfile enthält zunächst die erlaubte Transformationsfolge in Form eines dsig:Transforms Elements. Erscheint dieses dsig:Transforms Element genau wie angegeben in der korrespondierenden dsig:Reference, ist die Bedingung der übereinstimmenden Transformationsfolgen erfüllt.
            - Verwendet die mittels dsig:Transforms Element angegebene erlaubte Transformationsfolge implizite Transformationsparameter (vgl. die Definition in Abschnitt 2.2.2, Unterabschnitt *Implizite Transformationsparameter* in [SecLayer1.2]), müssen alle diese impliziten Transformationsparameter im Anschluss an das dsig:Transforms Element in VerifyTransformsInfoProfile angegeben werden, und zwar für jeden impliziten Transformationsparameter jeweils ein TransformParameter Element. Dabei stehen drei Möglichkeiten zur Auswahl:
                - TransformParameter/@URI bezeichnet den impliziten Transformationsparameter in exakt jener Weise, wie er in der zu prüfenden Signatur gebraucht wird (beispielsweise in der import-Direktive des Stylesheets einer XSLT-Transformation); Kindelemente für TransformParameter werden keine angegeben: Diese Variante soll der Client von MOA SP dann wählen, wenn MOA SP den impliziten Transformationsparamter von dem in URI angegebenen Ort herunterladen kann, und der Client die Hoheit über diesen Ort hat.
                - Wie oben, jedoch wird der implizite Transformationsparameter als Kind Base64Content von TransformParameter angegen: Diese Variante soll der

Client von MOA SP dann wählen, wenn MOA SP den impliziten Transformationsparamter nicht von dem in URI angegebenen Ort herunterladen kann.

- Wie oben, jedoch wird der Hashwert des impliziten Transformationsparameters als Kind Hash von TransformParameter angegen: Diese Variante soll der Client von MOA SP dann wählen, wenn MOA SP zwar den impliziten Transformationsparamter von dem in URI angegebenen Ort herunterladen kann, der Client aber nicht die Hoheit über diesen Ort hat.

5.1.2.1.3 ReturnHashInputData

Wird das optionale Element VerifyXMLSignatureRequest/ReturnHashInputData angegeben, liefert MOA SP in der Befehlsantwort für die geprüfte Signatur sowohl für jede dsig:Reference aus dsig:SignedInfo, als auch für jede dsig:Reference aus einem dsig:Manifest, auf das mittels des Attributs Type="http://www.w3.org/2000/09/xmldsig#Manifest" in einer dsig:Reference aus dsig:SignedInfo verwiesen wird, die Hash-Eingangsdaten zurück. Die Hash-Eingangsdaten sind jene Daten, die als Input für die Berechnung des Hashwertes der dsig:Reference verwendet werden.

Für die Spezifikation der Hash-Eingangsdaten in der Befehlsantwort siehe Abschnitt 5.1.3.1.2.

5.1.2.1.4 Wurzelzertifikate

Das Element <TrustProfileID> (als Child von <VerifyXMLSignatureRequest>) wird verwendet um ein Profil mit vertrauenswürdigen Wurzelzertifikaten zu selektieren. Profile werden im Zuge der Konfiguration von MOA definiert.

5.1.2.1.5 Supplements

Supplements können sowohl explizit (<SupplementProfile>) als auch über eine Profil-Id (<SupplementProfileID>) spezifiziert werden. Es können sowohl mehrere explizite als auch mehrere Profile angegeben werden. Ein Profil kann mehrere Supplements beinhalten.

5.1.2.1.6 SignatureInfo

<SignatureInfo>, <SignatureEnvironment>, <SignatureLocation> wurde in <VerifySignatureInfo>, <VerifySignatureEnvironment>, <VerifySignatureLocation> umbenannt, damit Elemente mit gleichem Namen immer unterschiedliche Inhalte beinhalten.

5.1.3 Antwort

Die Antwort orientiert sich an dem Message-Format für <VerifyXMLSignatureResponse> des Security-Layer.

5.1.3.1 Abweichung vom Security-Layer

5.1.3.1.1 Behördenkennzeichen

Ist das Zertifikat einer Behörde zugeordnet, d.h. das Zertifikat enthält nach [BehEig] eine Erweiterung mit OID 1.2.40.0.10.1.1.1, dann werden folgende Informationen über die Behörde retourniert:

- <PublicAuthority> (child von <SignerInfo>/<X509Data>)
- <Code> (child von <PublicAuthority>) enthält das Behördenkennzeichen (String)

wenn es vorhanden ist

5.1.3.1.2 Abdeckung der Signatur

Wurde in der Anfrage das Element ReturnHashInputData angegeben, liefert MOA SP in der Befehlsantwort für die geprüfte Signatur sowohl für jede dsig:Reference aus dsig:SignedInfo, als auch für jede dsig:Reference aus einem dsig:Manifest, auf das mittels des Attributs Type="http://www.w3.org/2000/09/xmldsig#Manifest" in einer dsig:Reference aus dsig:SignedInfo verwiesen wird, die Hash-Eingangsdaten als je ein HashInputData Element zurück (vergleiche Abschnitt 5.1.2.1.2).

Stammen die Hash-Eingangsdaten von einer dsig:Reference aus dsig:SignedInfo, enthält das Attribut HashInputData/@PartOf den Wert SignedInfo. Stammen die Hash-Eingangsdaten hingegen von einer dsig:Reference aus einem dsig:Manifest, enthält das Attribut HashInputData/@PartOf den Wert XMLDSIGManifest; weiters gibt in diesem Fall das Attribut HashInputData/@ReferringSigReference die Nummer jener dsig:Reference aus dsig:SignedInfo als positive Ganzzahl an, die auf das beinhaltende dsig:Manifest verweist.

Die Reihenfolge der HashInputData Elemente entspricht einerseits der Reihenfolge der dsig:Reference Elemente aus dsig:SignedInfo bzw. dsig:Manifest, andererseits werden zuerst die HashInputData Elemente für die dsig:Reference Elemente aus dsig:SignedInfo, und danach die HashInputData Elemente für die dsig:Reference Elemente aus allen referenzierten dsig:Manifest Elementen geliefert.

Die Daten in HashInputData sind entweder Base64- (Base64Content) oder XML-kodiert (XMLContent). Die Entscheidung, ob Daten als XML kodiert werden, trifft MOA SP (z.B. auf Grund von Informationen des http-Protokolls, wenn MOA SP eine http-Referenz auflöst).

5.1.3.1.3 Referenz-Eingangsdaten

Wurde in der Anfrage das Element SignatureManifestCheckParams angegeben, liefert MOA SP in der Befehlsantwort für die geprüfte Signatur sowohl für jede dsig:Reference aus dsig:SignedInfo, als auch für jede dsig:Reference aus einem dsig:Manifest, auf das mittels des Attributs Type="http://www.w3.org/2000/09/xmldsig#Manifest" in einer dsig:Reference aus dsig:SignedInfo verwiesen wird, die Referenz-Eingangsdaten als je ein ReferenceInputData Element zurück (vergleiche Abschnitt 5.1.2.1.2).

Stammen die Referenz-Eingangsdaten von einer dsig:Reference aus dsig:SignedInfo, enthält das Attribut ReferenceInputData/@PartOf den Wert SignedInfo. Stammen die Referenz-Eingangsdaten hingegen von einer dsig:Reference aus einem dsig:Manifest, enthält das Attribut ReferenceInputData/@PartOf den Wert XMLDSIGManifest; weiters gibt in diesem Fall das Attribut ReferenceInputData/@ReferringSigReference die Nummer jener dsig:Reference aus dsig:SignedInfo als positive Ganzzahl an, die auf das beinhaltende dsig:Manifest verweist.

Die Reihenfolge der ReferenceInputData Elemente entspricht einerseits der Reihenfolge der dsig:Reference Elemente aus dsig:SignedInfo bzw. dsig:Manifest, andererseits werden zuerst die ReferenceInputData Elemente für die dsig:Reference Elemente aus dsig:SignedInfo, und danach die ReferenceInputData Elemente für die dsig:Reference Elemente aus allen referenzierten dsig:Manifest Elementen geliefert.

Die Daten in ReferenceInputData sind entweder Base64- (Base64Content) oder XML-kodiert (XMLContent). Die Entscheidung, ob Daten als XML kodiert werden, trifft MOA SP (z.B. auf Grund von Informationen des http-Protokolls, wenn MOA SP eine http-Referenz auflöst).

Achtung: Die in den einzelnen ReferenceInputData Elementen enthaltenen Daten sind nur dann als vertrauenswürdig anzusehen, wenn MOA die Prüfungen laut Abschnitt 5.1.2.1.2 erfolgreich durchführen konnte. Dies ist dann der Fall, wenn das Element SignatureManifestCheck der Befehlsantwort den Code 0 enthält (vergleiche auch Abschnitt

5.1.3.1.4).

5.1.3.1.4 Überprüfung des SignaturManifest

Die Befehlsantwort enthält in `SignatureManifestCheck/Code` das Ergebnis der Prüfung des Siganturmanifests. Abhängig davon, ob in der Befehlsanfrage das Element `SignatureManifestCheckParams` angegeben wurde, ist die Bedeutung der zurückgelieferten Codes unterschiedlich:

- Wurde `SignatureManifestCheckParams` nicht angegeben, so ergibt sich die Bedeutung der Prüfcodes wie folgt:

Code	Bedeutung
0	Dieser Code hat eine der folgenden Bedeutungen: • Für diese Signatur ist kein Signaturmanifest notwendig. • Die Signatur enthält eine Referenz auf das notwendige Signaturmanifest. Das Signaturmanifest entspricht vom Umfang her den Anforderungen dieser Spezifikation. Für jede `dsig:Reference` des Signaturmanifests konnte der Hash-Wert erfolgreich überprüft werden.
2	Die Signatur enthält keine Referenz auf das notwendige Signaturmanifest.
3	Die Signatur enthält zwar eine Referenz auf das Signaturmanifest, dieses entspricht vom Umfang her jedoch nicht den Anforderungen aus [SecLayer1.2]. Die Hash-Werte der im Signaturmanifest vorhandenen `dsig:Reference` Elemente wurden nicht überprüft.
4	Die Signatur enthält eine Referenz auf das Signaturmanifest. Das Signaturmanifest entspricht vom Umfang her den Anforderungen aus [SecLayer1.2]. Bei der Überprüfung des Hash-Werts zumindest einer `dsig:Reference` des Signaturmanifests ist jedoch ein Fehler aufgetreten.

- Wurde `SignatureManifestCheckParams` angegeben, so ergibt sich die Bedeutung der Prüfcodes wie folgt:

Code	Bedeutung
0	Dieser Code hat eine der folgenden Bedeutungen: • Alle Referenzen halten die in der Befehlsanfrage zur Überprüfung der XML-Signatur gemachten Einschränkungen bezüglich der erlaubten Transformationsfolgen ein. Für diese Signatur ist kein Signaturmanifest notwendig. • Alle Referenzen halten die in der Befehlsanfrage zur Überprüfung der XML-Signatur gemachten Einschränkungen bezüglich der erlaubten Transformationsfolgen ein. Die Signatur enthält eine Referenz auf das notwendige Signaturmanifest. Das Signaturmanifest entspricht vom Umfang her den Anforderungen dieser Spezifikation. Für jede `dsig:Reference` des Signaturmanifests konnte der Hash-Wert erfolgreich überprüft werden.
1	Zumindest eine Referenz hält die in der Befehlsanfrage zur Überprüfung der XML-Signatur gemachten Einschränkungen bezüglich der erlaubten Transformationsfolgen nicht ein.
2	Alle Referenzen halten die in der Befehlsanfrage zur Überprüfung der XML-Signatur gemachten Einschränkungen bezüglich der erlaubten Transformationsfolgen ein. Die Signatur enthält keine Referenz auf das notwendige Signaturmanifest.
3	Alle Referenzen halten die in der Befehlsanfrage zur Überprüfung der XML-

	Signatur gemachten Einschränkungen bezüglich der erlaubten Transformationsfolgen ein. Die Signatur enthält zwar eine Referenz auf das Signaturmanifest, dieses entspricht vom Umfang her jedoch nicht den Anforderungen aus [SecLayer1.2]. Die Hash-Werte der im Signaturmanifest vorhandenen `dsig:Reference` Elemente wurden nicht überprüft.
4	Alle Referenzen halten die in der Befehlsanfrage zur Überprüfung der XML-Signatur gemachten Einschränkungen bezüglich der erlaubten Transformationsfolgen ein. Die Signatur enthält eine Referenz auf das Signaturmanifest. Das Signaturmanifest entspricht vom Umfang her den Anforderungen aus [SecLayer1.2]. Bei der Überprüfung des Hash-Werts zumindest einer `dsig:Reference` des Signaturmanifests ist jedoch ein Fehler aufgetreten.

5.1.3.2 Fehlerfälle/-codes

Fehlercodes werden retourniert wenn ein Kommando nicht ordnungsgemäß bearbeitet werden konnte. Der Mechanismus von SOAP FaultCodes wird verwendet um Fehlercodes zu retournieren. Nähere Details sind im Abschnitt zum SOAP-Binding enthalten.

Die Struktur der Fehlercodes wird vom Security-Layer übernommen und an die Anforderungen von MOA adaptiert:

Fehlerklassen

Die erste Ziffer enthält ein generisches Klassenschema:

1xxx	Fehler in Anfrage
2xxx	Fehler im MOA Modul
3xxx	Nicht anwendbar
4xxx	Fehler die mit dem Hardware-Token (HSM) in Verbindung stehen
5xxx	Fehler die bei Netzwerkkommunikation entstehen
9xxx	nicht klassifizierter Fehler

Funktionsklassen

Die zweite Ziffer enthält ein Klassenschema nach Unterscheidung der Funktionalität:

x1xx	Nicht anwendbar
x2xx	Modulfunktion
x3xx	Nicht anwendbar
x4xx	Zugriffsschutz einzelner Funktionen oder Daten (Authentisierung)
x5xx	Nicht anwendbar
x9xx	nicht klassifizierter Fehler

Zumindest die folgenden Fehlerfälle müssen geeignete Fehlercodes zugeordnet werden:

- Fehler in XML Struktur
- Base64 Kodierung fehlerhaft
- MOA unterstützt angegebenen Transformationsalgorithmus nicht
- Fehler bei Transformation aufgetreten (z.B. ungültiges Stylesheet,...)
- SignatureVerify: MOA unterstützt angegebene Signaturalgorithmus nicht
- Gewählter Schlüssel unterstützt angegebene Operation nicht (Signieren, DH, ...)
- Zugriff verweigert aufgrund von Zugriffpolicies
- Keine Zugriffsrechte auf privaten Schlüssel
- Kommando nicht implementiert
- Interner Fehler in MOA
- HSM nicht bereit (bzw. HSM nicht vorhanden)

- Interner HSM-Fehler (z.B. bei Berechnung)
- Angegebene Referenz (bei SignatureRequest) konnte nicht aufgelöst werden
- Implizite Transformationsreferenz (z.B. inkludierter Stylesheet) konnte nicht aufgelöst werden
- Auf eine notwendige Netzwerkressource konnte nicht zugegriffen werden
  - LDAP
- CRL nicht verfügbar, ungültig
- Fehler in der TLS Authentisierung

5.1.4 Funktionsbeschreibung

Die folgende Tabelle beschreibt den Ablauf der Signaturüberprüfung. Die Spalte „**SL**“ zeigt welche Schritte bei der Überprüfung nach Security-Layer Spezifikation durchgeführt werden. Die Spalte „**XMLDSig**“ zeigt welche Schritte sonst durchgeführt werden.

		SL	XMLDSig
1	**Prüfung der Signatur**		
1.1	Prüfung der <DigestValues> aller <Reference>n (core validation)	x	x
1.2	Prüfung der Signatur <SignatureValue> (core validation)	x	x
1.3	Prüfung des Signaturattributes <etsi:SigningCertificate>	x	
2	**Prüfung des/der Manifest(e)**		
2.1	Es wird <dsig:DigestValue> in allen <dsig:Reference>-Elementen in allen XMLDSig-Manifesten (siehe [SecLayer1.1], Abschnitt 3.2.2, Prüfung weiterer Manifeste) überprüft.	x	x
2.2	Es wird <dsig:DigestValue> in allen <dsig:Reference>-Elementen des Signaturmanifests (siehe [SecLayer1.1], Abschnitt 3.2.2, Prüfung des Signaturmanifests) überprüft.	x	
3	**Prüfung des Zertifikatsstatus**		
3.1	Aufbau des Zertifikatspfades Die Überprüfung des Zertifikatsstatus MUSS RFC 3280 entsprechen. Weiters müssen die Zertifikatspfade alle akkreditierten österreichischen ZDA unterstützt werden. In beiden Fällen sind ausschließlich folgende Erweiterungen bei der Validierung zu berücksichtigen: 1. AuthorityKeyIdentifier 2. SubjectKeyIdentifier 3. KeyUsage 4. ExtendedKeyUsage 5. BasicConstraints 6. CRLDistributionPoints Intermediate Zertifikate müssen für die Überprüfung in der folgenden Abfolge herangezogen werden: 1. Zertifikate, die in MOA gecached sind 2. Zertifikate, die in der Signatur mitgegeben wurden 3. Zertifikate, die aus dem Verzeichnis (via LDAP v3) geladen werden.	x	x
3.2	Prüfung aller Zertifikatsgültigkeiten Es wird sowohl das Ketten- als auch das Schalenmodell unterstützt.	x	x
3.3	Prüfung ob alle Zertifikate verwendbar sind • Keine unbekannten kritischen Erweiterungen • Richtig gesetzte BasicConstraints • Richtig gesetzte KeyUsage/ExtendedKeyUsage	x	x
3.4	Prüfung des Status aller Zertifikate mittels CRL CRLs müssen für die Überprüfung in der folgenden Abfolge herangezogen werden: 1. CRLs, die in MOA gecached sind	x	x

	2. CRLs, die in der Signatur mitgegeben wurden 3. CRLS, die aus dem Verzeichnis (via LDAP v3) geladen werden; Delta CRLS/OCSP wird derzeit nicht unterstützt (siehe *Abschnitt 10 Anhang – Nicht-Spezifizierte Funktionalität*)		
3.5	Überprüfung der Behörden-Erweiterung (non-critical)	x	x
3.6	Überprüfung ob es sich um ein Qualifiziertes Zertifikat handelt	x	x

5.1.5 Schnittstellendefinition

Das Interface besteht aus einer Request- und Reponse-Message im XML Format. Das XML Schema ist in [MOASchema] beschrieben.

Ein SOAP-basiertes Interface und ein API-basiertes Interface ist in *Abschnitt 8 Schnittstellendefinition* beschrieben.

5.1.6 Anforderungen an das Signaturformat

Das zu prüfende Signaturformat soll der Spezifikation des Security-Layers entsprechen. Ist das nicht der Fall, dann müssen folgende Anforderungen erfüllt sein:

<SignatureMethod>	Der Wert des Algorithm Attributs muss einer der folgenden sein: • http://www.w3.org/2000/09/xmldsig#rsa-sha1 • http://www.buergerkarte.at/namespaces/ecdsa/20020630# • http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha1 • http://www.w3.org/2001/04/xmldsig-more#rsa-md5"; • http://www.w3.org/2001/04/xmldsig-more#rsa-md2"; • http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"; • http://www.w3.org/2001/04/xmldsig-more#rsa-sha384"; • http://www.w3.org/2001/04/xmldsig-more#rsa-sha512"; • http://www.w3.org/2000/09/xmldsig#dsa-sha1"; • http://www.w3.org/2001/04/xmldsig-more#rsa-ripemd128"; • http://www.w3.org/2001/04/xmldsig-more/rsa-ripemd160"; • http://www.w3.org/2001/04/xmldsig-more#rsa-ripemd160"; • http://www.w3.org/2007/05/xmldsig-more#rsa-whirlpool"; • http://www.buergerkarte.at/namespaces/ecdsa/20020630#ecdsa-sha1"; • http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha224"; • http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256"; • http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha384"; • http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512"; • http://www.w3.org/2007/05/xmldsig-more#ecdsa-ripemd160"; • http://www.w3.org/2007/05/xmldsig-more#ecdsa-whirlpool";
<DigestMethod>	• http://www.w3.org/2000/09/xmldsig#sha1 • http://www.w3.org/2001/04/xmldsig-more#sha224 • http://www.w3.org/2001/04/xmlenc#sha256 • http://www.w3.org/2001/04/xmldsig-more#sha384 • http://www.w3.org/2001/04/xmlenc#sha512 • http://www.w3.org/2001/04/xmldsig-more#md5 • http://www.w3.org/2001/04/xmldsig-more#md2 • http://www.w3.org/2000/09/xmldsig-more#ripemd128 • http://www.w3.org/2001/04/xmlenc#ripemd160
<CanonicalizationMethod>	• http://www.w3.org/TR/2001/REC-xml-c14n-20010315 • http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments • http://www.w3.org/2001/10/xml-exc-c14n# • http://www.w3.org/2001/10/xml-exc-c14n#WithComments
<Transform>	Alle in [XMLDSig] und [ExcC14N] genannten Transformationen (Exclusive Canonical XML w/o comments, Exclusive Canonical XML

	w/ comments Canonical XML w/o comments, Canonical XML w/ comments, Base64, XSLT, XPath, Enveloped Signature) und XML-Signature XPath Filter 2.0 [XPathFilter2] wird unterstützt.
<KeyInfo>	Es muss das <KeyInfo> Element vorhanden sein. Mindestens darin enthalten sein muss das Signatorzertifikat oder ein Verweis darauf. Weiters angegeben sein dürfen weitere Zertifikate (oder Verweise darauf) zur Bildung der Zertifkatskette, sowie CRLs für die erwähnten Zertifikate. Zur Kodierung dieser Informationen dürfen die Kindelemente <X509Data> und <RetrievalMethod> verwendet werden. Ein <X509Data>-Element darf die Kindelemente <X509Certificate> und <X509CRL> enthalten. Ein <RetrievalMethod>-Element darf entweder auf ein <X509Data>-Element oder auf ein DER-kodiertes X509-Zertifikat verweisen.

5.1.7 Anforderungen an das Zertifikatsformat

Es werden alle X509 v3 Zertifikate nach RFC 3280 akzeptiert, die mit sha1RSA oder md5RSA signiert wurden.
Für die in *Abschnitt 5.1.4 Funktionsbeschreibung* berücksichtigten Erweiterungen gelten folgende Anforderungen:

AuthorityKeyIdentifier	Optional
SubjectKeyIdentifier	Optional
KeyUsage	Optional
ExtendedKeyUsage	Optional
BasicConstraints	Optional
CRLDistributionPoints	Optional

5.1.8 Archivierung von CRLs

CRLs, die einmal ins System gelangt sind, werden archiviert, um die spätere Prüfung zu einem in der Vergangenheit liegenden Zeitpunkt zu erleichtern. Wie lange CRLs archiviert werden kann konfiguriert werden.

5.2 Signaturprüfung nach CMS

5.2.1 Zielsetzung

Es müssen sowohl CMS Signaturen, die der Security-Layer Spezifikation (Version 1.1) [SecLayer1.1] entsprechen, als auch Signaturen, die dem RFC 2630 entsprechen überprüft werden. Einschränkungen an das Signaturformat für letzteren Fall werden in *Abschnitt 5.2.6* Anforderungen an das Signaturformat spezifiziert.

5.2.2 Anfrage

Die Anfrage orientiert sich an dem Messageformat für <VerifyCMSSignatureRequest> des Security-Layer.

5.2.2.1 Abweichungen vom Security-Layer

5.2.2.1.1 Wurzelzertifikate

Siehe *Abschnitt 5.1.2.1.4 Wurzelzertifikate*.

5.2.3 Antwort

Die Antwort orientiert sich an dem Messageformat für <VerifyCMSSignatureResponse> des Security-Layer.

5.2.3.1 Abweichungen vom Security-Layer

5.2.3.1.1 Behördenkennzeichen

Siehe *Abschnitt 5.1.3.1.1* Behördenkennzeichen.

5.2.3.2 Fehlerfälle/-codes

Siehe *Abschnitt 5.1.3.2 Fehlerfälle/-codes*.

5.2.4 Funktionsbeschreibung

Die folgende Tabelle beschreibt den Ablauf der Signaturüberprüfung. Die Spalte „**SL**“ zeigt welche Schritte bei der Überprüfung nach Security-Layer Spezifikation durchgeführt werden. Die Spalte „**CMS**“ zeigt welche Schritte sonst durchgeführt werden.

		SL	CMS
1	**Prüfung der Signatur**		
1.1	Prüfung der Signatur	x	x
1.2	• signedAttributes werden nicht ausgewertet • von den unsignedAttributes werden nur Zertifikate und CRLs ausgewertet	x	x
1.3	von den signedAttributes wird OtherSigningCertificate ausgewertet	x	
2	**Prüfung des Zertifikatsstatus**		
2.1	Aufbau des Zertifikatspfades Die Überprüfung des Zertifikatsstatus MUSS RFC 3280 entsprechen. Weiters müssen die Zertifikatspfade alle akkreditierten österreichischen ZDA unterstützt werden. In beiden Fällen sind ausschließlich folgende Erweiterungen bei der Validierung zu berücksichtigen: 1. AuthorityKeyIdentifier 2. SubjectKeyIdentifier 3. KeyUsage 4. ExtendedKeyUsage 5. BasicConstraints 6. CRLDistributionPoints Intermediate Zertifikate müssen für die Überprüfung in der folgenden Abfolge herangezogen werden: 1. Zertifikate, die in MOA gecached sind 2. Zertifikate, die in der Signatur mitgegeben wurden 3. Zertifikate, die aus dem Verzeichnis (via LDAP v3) geladen werden	x	x
2.2	Prüfung aller Zertifikatsgültigkeiten Es wird sowohl das Ketten- als auch das Schalenmodell unterstützt.	x	x
2.3	Prüfung ob alle Zertifikate verwendbar sind • Keine unbekannten kritischen Erweiterungen • Richtig gesetzte BasicConstraints • Richtig gesetzte KeyUsage/ExtendedKeyUsage	x	x
2.4	Prüfung des Status aller Zertifikate mittels CRL CRLs müssen für die Überprüfung in der folgenden Abfolge herangezogen werden: 1. CRLs, die in MOA gecached sind 2. CRLs, die in der Signatur mitgegeben wurden 3. CRLS, die aus dem Verzeichnis (via LDAP v3) geladen werden; Delta CRLS/OCSP wird derzeit nicht unterstütz (siehe *Abschnitt 10 Anhang – Nicht-Spezifizierte Funktionalität*)	x	x
2.5	Überprüfung der Behörden-Erweiterung (non-critical)	x	x
2.6	Überprüfung ob es sich um ein Qualifiziertes Zertifikat handelt	x	x

5.2.5 Schnittstellendefinition

Siehe *Abschnitt 5.1.5 Schnittstellendefinition*.

5.2.6 Anforderungen an das Signaturformat

Das zu prüfende Signaturformat soll der Spezifikation des Security-Layers entsprechen. Ist das nicht der Fall, dann müssen folgende Anforderungen erfüllt sein:

Content Type	es wird nur Signed-Data Content Type unterstützt Signaturen können attached oder detached sein
digestAlgorithms	md5, sha-1
Certificates	zumindest das Signator-Zertifikat muss inkludiert sein
CRLs	die Angabe dieser CRLs ist nicht verpflichtend. Verwendung ist in 5.2.4 definiert.

5.2.7 Anforderungen an das Zertifikatsformat

Siehe *Abschnitt 5.1.7 Anforderungen an das Zertifikatsformat*.

5.2.8 Archivierung von CRLs

Siehe *Abschnitt 5.1.8 Archivierung von CRLs.*

6 Modul Server-Signatur

Dieses Modul dient zur Erstellung von XMLDSig und CMS bzw. XAdES und CAdES Signaturen.

6.1 CMS/CAdES Server-Signatur

6.1.1 Zielsetzung

Dieser Funktionsaufruf dient zum Erstellen von CMS Signaturen bzw. CAdES-Signaturen nach der Security-Layer Spezifikation [SecLayer1.2x].

Bei der Erstellung solcher Signaturen ist darauf Rücksicht zu nehmen, dass eine Rekonstruktion der auf Papier ausgedruckten Signatur in ihre elektronische Form einfach möglich ist. Das bedeutet, dass unabhängig vom Zeitpunkt der Erstellungsanfrage stets die gleiche Signatur an die Applikation zurückgeliefert werden muss ($Sig(Daten)_{t=t1} = Sig(Daten)_{t=t2}$). Beispielsweise ist der Einsatz von zufällig gewählten oder zeitabhängigen XML-IDs zu vermeiden.

Eine Signatur muss sowohl in Software als auch mittels eines HSMs erstellt werden können. Wird ein HSM verwendet, dann muss diese Schnittstelle PKCS #11 verwenden.

6.1.2 Anfrage

Die Antwort orientiert sich an dem Message-Format für `<CreateCMSSignatureRequest>` des Security-Layer.

6.1.2.1 Abweichungen vom Security-Layer

6.1.2.1.1 Erstellung einer Security-Layer konformen Signatur

Das Attribut `SecurityLayerConformity` spezifiziert ob eine Signatur gemäß der Security-Layer Spezifikation erzeugt werden soll. Ist `SecurityLayerConformity="true"` dann wird eine CAdES-Signatur entsprechend der Security-Layer Spezifikation erzeugt. Ist das Attribut `"false"` wird eine herkömmliche CMS Signatur erzeugt. Das Attribut ist optional, der Default-Wert ist `true`.

6.1.2.1.2 KeyIdentifier

`<KeyIdentifier>` spezifiziert eine Schlüsselgruppe aus der ein Schlüssel zur Signaturerstellung ausgewählt werden muss (vgl. `<KeyboxIdentifier>` des Security-Layer).

Eine Schlüsselgruppe ist eine Zusammenfassung von mehreren gleichwertigen Schlüsseln, die aus Sicht der Online-Applikation wie *ein* Schlüssel behandelt wird. Werden HSMs verwendet, dann können Schlüssel einer Schlüsselgruppe auf mehrere HSMs verteilt werden. Die Definition der Schlüsselgruppe erfolgt im Zuge der Konfiguration von MOA.

6.1.2.1.3 Stapelsignaturen

Stapelsignaturen werden durchgeführt indem mehrere „Dokumente“ mit einem Schlüssel signiert werden und daraus mehrere Signaturen (`<CMSSignature>`) erstellt werden.

Stapelsignaturen werden dadurch spezifiziert, dass das Element `<SingleSignatureInfo>` mehrfach auftreten kann.

6.1.3 Antwort

Die Antwort orientiert sich an dem Messageformat für <CreateCMSSignatureResponse> des Security-Layer.

6.1.3.1 Abweichungen vom Security-Layer

6.1.3.1.1 Stapelsignaturen

Für jedes <SingleSignatureInfo> Element des Aufrufs enthält <CreateCMSSignatureResponse> entweder ein <CMSSignature>-Element, falls die Signaturerstellung erfolgreich war, bzw. ein <ErrorResponse>-Element, falls die Signaturerstellung gescheitert ist. Die Reihenfolge der Antwortelemente entspricht der Reihenfolge der übergebenen <SingleSignatureInfo>s.

6.1.3.2 Fehlerfälle/-codes

Siehe *Abschnitt 5.1.3.2 Fehlerfälle/-codes*.

6.1.4 Funktionsbeschreibung

1	**Ermittlung des Signaturschlüssels** Basierend auf dem KeyIdentifier wird eine Gruppe von privaten Signaturschlüsseln identifiziert. Aus dieser Gruppe wird ein Schlüssel ausgewählt.
2	**Zertifikatsprüfung** Für den selektierten Schlüssel wird überprüft ob das dazugehörende Zertifikat noch gültig ist. Im Fehlerfall wird Schritt 1 wiederholt. Es wird im Logging erfasst, dass ein ungültiger Schlüssel vorliegt
3	**Erstellung der Signatur** Für jedes <SingleSignatureInfo> wird eine Signatur (<CMSSignature>) erstellt.

6.1.5 Interface-Design

6.1.5.1 XML Schema für Request-Response Message Format

Das Interface besteht aus einer Request- und Reponse-Message im XML Format. Das XML Schema ist in [MOASchema] beschrieben.

6.2 XMLDSig Server-Signatur nach Security-Layer

6.2.1 Zielsetzung

Dieser Funktionsaufruf dient zum Erstellen von XMLDSig Signaturen bzw. nach XAdES Signaturen nach der Security-Layer Spezifikation [SecLayer1.2x].
Bei der Erstellung solcher Signaturen ist darauf Rücksicht zu nehmen, dass eine Rekonstruktion der auf Papier ausgedruckten Signatur in ihre elektronische Form einfach möglich ist. Das bedeutet, dass unabhängig vom Zeitpunkt der Erstellungsanfrage stets die gleiche Signatur an die Applikation zurückgeliefert werden muss ($Sig(Daten)_{t=t1} = Sig(Daten)_{t=t2}$). Beispielsweise ist der Einsatz von zufällig gewählten oder zeitabhängigen XML-IDs zu vermeiden.

Eine Signatur muss sowohl in Software als auch mittels eines HSMs erstellt werden können. Wird ein HSM verwendet, dann muss diese Schnittstelle PKCS #11 verwenden.

6.2.2 Anfrage

Die Antwort orientiert sich an dem Message-Format für <CreateXMLSignatureRequest> des Security-Layer.

6.2.2.1 Abweichungen vom Security-Layer

6.2.2.1.1 Angabe von Daten

Für die Elemente SingleSignatureInfo/DataObjectInfo/DataObject, SingleSignatureInfo/DataObjectInfo/CreateTransformsInfoProfile/Supplement/Content, SingleSignatureInfo/CreateSignatureInfo/CreateSignatureEnvironment sowie SingleSignatureInfo/CreateSignatureInfo/CreateSignatureEnvironmentProfile/Supplement/Content steht neben den auch von der Security-Layer Spezifikation bekannen Möglichkeiten XMLContent und Base64Content die dritte Möglichkeit des Elements LocRefContent zur Verfügung. Als Inhalt von LocRefContent muss eine URL angegeben werden, die von MOA SS aufgelöst wird, um an die zu verwendenden Daten zu kommen.

6.2.2.1.2 Erstellung einer Security-Layer konformen Signatur

Das Attribut SecurityLayerConformity spezifiziert ob eine Signatur gemäß der Security-Layer Spezifikation erzeugt werden soll. Ist SecurityLayerConformity="true" dann wird eine XAdES Signatur entsprechend der Security-Layer Spezifikation erzeugt. Ist das Attribut "false" wird eine herkömmliche XMLDSig Signatur erzeugt. Das Attribut ist optional, der Default-Wert ist true.

6.2.2.1.3 KeyIdentifier

<KeyIdentifier> spezifiziert eine Schlüsselgruppe aus der ein Schlüssel zur Signaturerstellung ausgewählt werden muss (vgl. <KeyboxIdentifier> des Security-Layer).

Eine Schlüsselgruppe ist eine Zusammenfassung von mehreren gleichwertigen Schlüsseln, die aus Sicht der Online-Applikation wie *ein* Schlüssel behandelt wird. Werden HSMs verwendet, dann können Schlüssel einer Schlüsselgruppe auf mehrere HSMs verteilt werden. Die Definition der Schlüsselgruppe erfolgt im Zuge der Konfiguration von MOA.

6.2.2.1.4 Stapelsignaturen

Stapelsignaturen werden durchgeführt indem mehrere „Dokumente" mit einem Schlüssel signiert werden und daraus mehrere Signaturen (<dsig:Signature>) erstellt werden.

Stapelsignaturen werden dadurch spezifiziert, dass das Element <SingleSignatureInfo> mehrfach auftreten kann.

6.2.2.1.5 Profil für Transformationen

Bei <CreateXMLSignatureRequest> können die Daten von <CreateTransformsInfo> und den optionalen <Supplement>-Elementen durch die Spezifikation einer Profil-ID (<CreateTransformsInfoID>) ersetzt werden.

Profile für die Transformationen werden im Zuge der der Konfiguration von MOA definiert.

6.2.2.1.6 Profil für CreateSignatureEnvironment

Die Position, an der eine Signatur in ein XML Dokument eingefügt werden muss (<CreateSignatureLocation>), sowie optionale Supplements, können entweder explizit spezifiziert werden, oder mittels einer Profil-ID (<CreateSignatureEnvironmentProfileID>) referenziert werden.

6.2.2.1.7 ChildOfManifest

Das boolean Attribut ChildOfManifest in <CreateXMLSignatureRequest>/-<SingleSignatureInfo>/<DataObjectInfo> spezifiziert ob eine Referenz in die Signatur selbst, oder in ein XMLDSig-Manifest gegeben werden muss.

Weist zumindest ein <DataObjectInfo> pro Signatur dieses Attribut mit einem Wert true auf, erstellt MOA ein XML-DSig-Manifest, das alle mit diesem Attribut gekennzeichneten Referenzen enthält.

Ist das Attribut SecurityLayerConformity im übergeordneten Element <SingleSignatureInfo> auf den Wert true gesetzt, darf das Attribut ChildOfManifest nicht gesetzt sein bzw. nur den Wert false annehmen.

6.2.3 Antwort

Die Antwort orientiert sich an dem Messageformat für <CreateXMLSignatureResponse> des Security-Layer.

6.2.3.1 Abweichungen vom Security-Layer

6.2.3.1.1 Stapelsignaturen

Für jedes <SingleSignatureInfo> Element des Aufrufs enthält <CreateXMLSignatureResponse> entweder ein <SignatureEnvironment>-Element, falls die Signaturerstellung erfolgreich war, bzw. ein <ErrorResponse>-Element, falls die Signaturerstellung gescheitert ist. Die Reihenfolge der Antwortelemente entspricht der Reihenfolge der übergebenen <SingleSignatureInfo>s.

6.2.3.2 Fehlerfälle/-codes

Siehe *Abschnitt 5.1.3.2 Fehlerfälle/-codes*.

6.2.4 Funktionsbeschreibung

4	**Ermittlung des Signaturschlüssels** Basierend auf dem `KeyIdentifier` wird eine Gruppe von privaten Signaturschlüsseln identifiziert. Aus dieser Gruppe wird ein Schlüssel ausgewählt.
5	**Zertifikatsprüfung** Für den selektierten Schlüssel wird überprüft ob das dazugehörende Zertifikat noch gültig ist. Im Fehlerfall wird Schritt 1 wiederholt. Es wird im Logging erfasst, dass ein ungültiger Schlüssel vorliegt
6	**Erstellung der Signatur** Für jedes `<SingleSignatureInfo>` wird eine Signatur (`<SignatureEnvironment>`) erstellt.

6.2.5 Interface-Design

6.2.5.1 XML Schema für Request-Response Message Format

Das Interface besteht aus einer Request- und Reponse-Message im XML Format. Das XML Schema ist in [MOASchema] beschrieben.

7 Konfiguration von MOA

MOA muss folgende Konfigurationsmöglichkeiten bereitstellen, wobei die Konfiguration durch das Editieren von Konfigurations-Dateien erfolgen kann.

Die beschriebenen Konfigurationsmöglichkeiten können entweder für das Modul Signaturprüfung (SP) und/oder für das Modul Server-Signatur (SS) relevant sein.

7.1 Client-Applikationen

Jeder Client-Applikation müssen folgende Informationen zugeordnet werden können:

- ein TLS-Client Zertifikat, mit dem sich die Client-Applikation per TLS an MOA anmelden kann (Modul SS und SP)
- eine Menge von KeyIdentifiern, die zur Signaturerstellung verwendet werden können (Modul SS)

7.2 TLS-Authentisierung

Sowohl die TLS Client-Authentisierung als auch die Server-Authentisierung muss getrennt voneinander eingeschaltet bzw. ausgeschaltet werden können (Modul SP und SS).

7.3 Zertifizierungsdiensteanbieter (ZDA)

Ein ZDA kann mehrere Certificate Authorities (CA) betreiben. Jede dieser CAs stellt entweder Endanwender-Zertifikate oder Intermediate-CA Zertifikate aus. CRLs, die von diesen CAs erstellt werden, werden in einem Verzeichnisdienst publiziert.

Für CAs, die dem Betreiber von MOA bekannt sind, müssen CRL-Distribution Points definiert werden können (Modul SP und SS).

Dadurch kann auch der Status von Zertifikaten überprüft werden, ohne dass die Zertifikate die CRLDistributionPoint-Erweiterung beinhalten. Weiters können diese CRLs auch regelmäßig gecached werden, und damit die Zertifikatsüberprüfung beschleunigt werden.

Weiters müssen Distribution Points von Intermediate-CAs konfiguriert werden können, an denen die CA Zertifikate publiziert sind.

Zertifikate von CAs, die sich im System befinden, müssen gecached werden.

7.4 Private Schlüssel in der DB

Private Schlüssel, die nicht in einem HSM verwaltet werden, müssen in Form eines PKCS#12 Files in MOA importiert werden (Modul SS).

7.5 Zertifikate für öffentliche Schlüssel

Für alle Zertifikate von privaten Schlüsseln, die in einem HSM verwaltet werden und die in einer DB verwaltet werden, müssen Zertifikate gespeichert werden (Modul SS).

7.6 KeyIdentifier

Allen KeyIdentifiern müssen 1 oder mehrere private Schlüssel (eine Schlüsselgruppe) zugeordnet werden (Modul SS).

7.7 Profile

Allen Profilen (IDs) müssen die eigentlichen Daten zugeordnet werden:

Profilbezeichner	Zugeordnete Daten	Modul
<VerifyTransformsInfoProfileID>	<VerifyTransformsInfoProfile>	SP
<TrustProfileID>	eine Menge von vertrauens-würdigen Zertifikaten	SP
<CreateTransformsInfoID>	<CreateTransformsInfo> und optional <Supplement>s	SS
<CreateSignatureEnvironmentProfileID>	<CreateSignatureLocation> und optionale <Supplement>s	SS
<SupplementProfileID>	mehrere <SupplementProfile>-Elemente	SP

7.8 Algorithmen für XMLDSig

URIs zur Spezifikation von Algorithmen für XMLDSig, die in MOA implementiert sind, müssen konfigurierbar sein (Modul SP und SS).

7.9 CRL

Es kann konfiguriert werden, wie lange CRLs in MOA archiviert werden müssen.

8 Schnittstellendefinition

Die spezifizierte Funktionalität kann sowohl über SOAP (für Aufrufe über das Netzwerk) als auch über API Aufrufe verwendet werden.

8.1 SOAP

[MOA WSDL] beschreibt die MOA-Schnittstelle bei Verwendung als Web-Service mit SOAP als Transportprotokoll. Aus dem WSDL-Dokument können mit geeigneten Tools Client-Stub- und Server-Tie-Objekte generiert werden.

Die SOAP Spezifikation basiert auf SOAP Version 1.1 [SOAP], die WSDL Spezifikation auf WSDL Version 1.1 [WSDL].

Das Dokument folgt dem hierarchischen Aufbau von WSDL-Dokumenten (siehe [WSDL] Abschnitt 1).

- Types: Die dem Web-Service zugrundeliegenden Datentypen sind im XML-Schema von MOA enthalten, und werden aus diesem importiert.
- Messages: Die Request- und Response-Objekte aus dem MOA Schema werden als Messages gekapselt, um den Operationen als Eingangs- bzw. Rückgabewerte zur Verfügung zu stehen. Für SOAP Fehlermeldungen wird der Message-Typ `MOAFault` verwendet.
- Port Types: Die Messages werden in 2 Port Types zusammengefasst:
  - SignatureCreationPortType enthält die Operationen zum Erstellen von Signaturen auf Basis von CMS und XML
  - SignatureVerificationPortType enthält die Operationen zum Prüfen von Signaturen auf Basis von XML
- Binding: Für die beiden Ports wird jeweils ein dokument-orientiertes SOAP-Binding definiert, das den Aufbau der SOAP-Messages spezifiziert. Ein dokument-orientiertes Binding wurde einem RPC-Binding vorgezogen, weil es den XML-Content der Parameter und Rückgabewerte besser verdeutlicht.
- Service: Die beiden Bindings werden separiert als Service angeboten.

8.1.1 Beispiele für SOAP Messages

8.1.1.1 Request

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <moa:CreateXMLSignatureRequest
       xmlns:moa="http://reference.e-government.gv.at/namespace/moa/20020822#">
      ...
    </moa:CreateXMLSignatureRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

8.1.1.2 Response

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <moa:CreateXMLSignatureResponse
       xmlns:moa="http://reference.e-government.gv.at/namespace/moa/20020822#">
      ...
    </moa:CreateXMLSignatureResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

8.1.1.3 Response im Fehlerfall

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Server</faultcode>
      <faultstring>Server Error</faultstring>
      <detail>
        <moa:ErrorResponse
       xmlns:moa="http://reference.e-government.gv.at/namespace/moa/20020822#">
          <ErrorCode>2000</ErrorCode>
          <Info>Fehler im MOA Modul</Info>
        </moa:ErrorResponse>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

8.2 API

Die in [MOASchema] definierten Message-Formate sind in geeigneten Java Interfaces modelliert. Die Art der Modellierung orientiert sich an der im Package `javax.xml.soap` definierten Interfaces ohne diese jedoch direkt zu referenzieren.

Die Server-seitige Schnittstelle ist durch die Interfaces `SignatureCreationService` und `SignatureVerificationService` definiert.

Die Definitionen der weiteren verwendeten Interfaces sind in [MOA API] dokumentiert.

8.2.1 Interface SignatureCreationService

```
package at.gv.egovernment.moa;

/**
 * Interface providing functions for signature creation.
 */
public interface SignatureCreationService {
  /**
   * Create an XML signature.
   */
  public CreateXMLSignatureResponse
           createXMLSignature(CreateXMLSignatureRequest request)
           throws MOAException;
}
```

8.2.2 Interface SignatureVerificationService

```
package at.gv.egovernment.moa;

/**
 * Interface providing functions for verifying signatures.
 */
public interface SignatureVerificationService {
  /**
   * Verify the given CMS signature.
   */
  public VerifyCMSSignatureResponse
           verifyCMSSignature(VerifyCMSSignatureRequest request)
           throws MOAException;
  /**
   * Verify the given XML signature.
   */
  public VerifyXMLSignatureResponse
           verifyXMLSignature(VerifyXMLSignatureRequest request)
           throws MOAException;
}
```

9 Referenzen

Referenz	Titel	Verfasser	Datum
[UC CIO]	Use Cases für MOA	CIO	31.07.2002
[UC BRZ]	Anwendungsfälle Spezifikation	BRZ	9.07.2002
[SecLayer1.1]	Schnittstellenspezifikation Security-Layer Version 1.1 http://www.buergerkarte.at/konzept/securitylayer/spezifikation/20020831/core/Core.html	CIO	31.08.2002
[SecLayer1.2]	Schnittstellenspezifikation Security-Layer Version 1.2 http://www.buergerkarte.at/konzept/securitylayer/spezifikation/20040514/core/Core.html	CIO	14.05.2004
[SecLayer1.2x]	Schnittstellenspezifikation Security-Layer Version 1.2x http://www.buergerkarte.at/konzept/securitylayer/spezifikation/20040514/core/Core.html	EGIZ	xx.xx.2013
[XMLDSig]	XML-Signature Syntax and Processing	W3C	12.02.2002
[QCPro]	Qualified Certificates Profile, IETF RFC 3039	IETF	01.2001
[ECDSAXML]	ECDSA with XML-Signature Syntax http://www.ietf.org/internet-drafts/draft-blake-wilson-xmldsig-ecdsa-09.txt	IETF	03.2004
[ExcC14N]	Exclusive XML Canonicalization, V 1.0	W3C	18.07.2002
[XPathFilter2]	XML-Signature XPath Filter 2.0	W3C	8.11.2002
[CMS]	Cryptographic Message Syntax, IETF RFC 2630	IETF	06.1999
[MOASchema]	Datei MOA-SPSS-1.5.2.xsd	ARGE/EGIZ	15.05.2013
[MOA WSDL]	Datei MOA-SPSS-1.5.2.wsdl	ARGE/EGIZ	15.05.2013
[MOA API]	Java Package `at.gv.egovernment.moa`	ARGE	23.08.2002
[BehEig]	Object Identifier und Einsatz in X.509 Zertifikaten V 1.0.1	CIO	6.08.2002
[TLS]	The TLS Protocol Version 1.0, IETF RFC 2046	IETF	01.1999
[WSDL]	Web Services Description Language (WSDL) 1.1	W3C	15.03.2001
[SOAP]	Simple Object Access Protocol (SOAP) 1.1	W3C	8.05.2000

10 Anhang – Nicht-Spezifizierte Funktionalität

Dieser Abschnitt beschreibt welche Funktionalität dzt. nicht spezifiziert wurde, warum diese nicht spezifiziert wurde, und dass dies in einer folgenden Version zu spezifizieren ist.

10.1 Standards

Es sind zusätzlich zu den bereits aufgelisteten Standards keine weiteren (ausländischen) Standards zu berücksichtigen.

10.2 Signaturprüfung

10.2.1 OCSP

OCSP wird derzeit nicht unterstützt, weil kein österreichscher akkreditierter ZDA produktive OCSP Responder einsetzt. Damit ist auch die Unterstützung der Zertifikatserweiterung `AuthorityInformationAccess` nicht notwendig.

10.2.2 Delta CRLs

Delta CRLs werden derzeit nicht unterstützt, weil kein österreichscher akkreditierter ZDA produktive Delta CRLs unterstützt. Damit ist auch die Unterstützung der Zertifikatserweiterungen `FreshestCRL` nicht notwendig.

10.2.3 Zertifikatserweiterungen

Es werden keine Zertifikatserweiterungen unterstützt, die sich auf die Handhabung von „Certificate Policies“ bzw. „Naming Constraints“ beziehen.

Damit wird Cross-Zertifizierung derzeit nicht unterstützt (kein österreichscher akkreditierter ZDA unterstützt Cross-Zertifizierung).

Zertifikate, die eine `NameConstraints` Extension aufweisen und diese laut RFC 3280 `critical` gesetzt ist, können nicht akzeptiert werden.

Eine Spezifikation und Implementierung dieser Funktionalität ist erst dann sinnvoll, wenn es Zertifikatsprodukte gibt, die diese Erweiterungen verwenden.