Minimale Umsetzung der Applikationsschnittstelle Security-Layer zur österreichischen Bürgerkarte		Konvention 1.2.1 Empfehlung
Kurzbeschreibung	Dieses Dokument beschreibt folgende Anforderungen an eine minimale Umsetzung der Schnittstelle *Security-Layer* durch eine *Bürgerkarten-Umgebung*: • Schnittstellenbefehle • Transportprotokolle • Profil für die Erstellung und Prüfung von CMS-Signaturen • Profil für CMS-Verschlüsselung und CMS-Entschlüsselung • Profil für XML-Verschlüsselung und XML-Entschlüsselung • Profil für Hashwert-Berechnung und Hashwert-Verifikation • Anzeigeformate und Zeichensätze	
Autoren:	Arno Hollosi Gregor Karlinger Thomas Rössler Martin Centner et al.	Projektteam/Arbeitsgruppe AG Bürgerkarte
Datum:	20.2.2008	

Inhaltsverzeichnis

1. Einführung

Dieses Dokument beschreibt die Anforderungen an eine minimale Umsetzung der Schnittstelle *Security-Layer* durch eine *Bürgerkarten-Umgebung*. Möchte eine *Bürgerkarten-Umgebung* konform zum Modell Bürgerkarte sein, MUSS sie alle in den nachfolgenden Kapiteln als ERFORDERLICH oder EMPFOHLEN gekennzeichneten Funktionalitäten umsetzen. Die Umsetzung von Funktionalitäten, die als EMPFOHLEN gekennzeichnet sind, darf in gut begründbaren Ausnahmefällen unterbleiben (vergleiche dazu die Bedeutung des Schlüsselworts EMPFOHLEN in [KEYWORDS]).

1.1. Namenskonventionen

Zur besseren Lesbarkeit wurde in diesem Dokument auf geschlechtsneutrale Formulierungen verzichtet. Die verwendeten Formulierungen richten sich jedoch ausdrücklich an beide Geschlechter.

Folgende Namenraum-Präfixe werden in dieser Spezifikation zur Kennzeichnung der Namensräume von XML-Elementen verwendet:

Präfix	Namenraum	Erläuterung
dsig	http://www.w3.org/2000/09/xmldsig#	Elemente aus [XMLDSIG]
dsm	http://www.w3.org/2001/04/xmldsig-more#	Elemente aus [XMLDSIG-URI] und [ECDSA-XML]
xenc	http://www.w3.org/2001/04/xmlenc#	Elemente aus [XMLEnc]
sl	http://www.buergerkarte.at/namespaces/securitylayer/1.2#	Elemente dieser Spezifikation

1.2. Schlüsselwörter

Dieses Dokument verwendet die Schlüsselwörter MUSS, DARF NICHT, ERFORDERLICH, SOLLTE, SOLLTE NICHT, EMPFOHLEN, DARF und OPTIONAL zur Kategorisierung der Anforderungen. Diese Schlüsselwörter sind analog zu ihren englischsprachigen Entsprechungen MUST, MUST NOT, REQUIRED, SHOULD, SHOULD NOT, RECOMMENDED, MAY, und OPTIONAL zu handhaben, deren Interpretation in [KEYWORDS] festgelegt ist.

2. Schnittstellenbefehle des Security-Layer

Die nachfolgende Tabelle listet alle Schnittstellenbefehle, die in *Applikationsschnittstelle Security-Layer* spezifiziert sind. Für jeden dieser Schnittstellenbefehle gibt sie an, ob der Befehl in einer minimalen Umsetzung der *Bürgerkarten-Umgebung* enthalten sein muss (Anforderungs-Level ERFORDERLICH oder EMPFOHLEN) oder nicht (Anforderungs-Level OPTIONAL).

Signaturbefehl	Kommandonamen	Anforderung
Signatur nach CMS erstellen	sl:CreateCMSSignatureRequest sl:CreateCMSSigantureResponse	EMPFOHLEN
Signatur nach XMLDSIG erstellen	sl:CreateXMLSignatureRequest sl:CreateXMLSignatureResponse	ERFORDERLICH
Signatur nach CMS prüfen	sl:VerifyCMSSignatureRequest sl:VerifyCMSSignatureResponse	EMPFOHLEN
Signatur nach XMLDSIG prüfen	sl:VerifyXMLSignatureRequest sl:VerifyXMLSignatureResponse	ERFORDERLICH
Verschlüsselung als CMS-Nachricht	sl:EncryptCMSRequest sl:EncryptCMSResponse	EMPFOHLEN
Verschlüsselung als XML-Nachricht	sl:EncryptXMLRequest sl:EncryptXMLResponse	EMPFOHLEN
Entschlüsselung einer CMS-Nachricht	sl:DecryptCMSRequest sl:DecryptCMSResponse	EMPFOHLEN
Entschlüsselung einer XML-Nachricht	sl:DecryptXMLRequest sl:DecryptXMLResponse	EMPFOHLEN
Hashwert-Berechnung	sl:CreateHashRequest sl:CreateHashResponse	ERFORDERLICH
Hashwert-Verifikation	sl:VerifyHashRequest sl:VerifyHashResponse	EMPFOHLEN
Abfrage verfügbarer Infoboxen	sl:InfoboxAvailableRequest sl:InfoboxAvailableResponse	ERFORDERLICH
Anlegen einer Infobox	sl:InfoboxCreateRequest sl:InfoboxCreateResponse	ERFORDERLICH
Löschen einer Infobox	sl:InfoboxDeleteRequest sl:InfoboxDeleteResponse	ERFORDERLICH
Lesen von Infobox-Daten	sl:InfoboxReadRequest sl:InfoboxReadResponse	ERFORDERLICH
Verändern von Infobox-Daten	sl:InfoboxUpdateRequest sl:InfoboxUpdateResponse	ERFORDERLICH
Null-Operation	sl:NullOperationRequest sl:NullOperationResponse	ERFORDERLICH
Abfrage der Umgebungseigenschaften	sl:GetPropertiesRequest sl:GetPropertiesResponse	ERFORDERLICH
Abfrage des Tokenstatus	sl:GetStatusRequest sl:GetStatusResponse	ERFORDERLICH

3. Transportprotokolle für den Security-Layer

Das Dokument Transportprotokolle Security-Layer beschreibt eine Reihe von möglichen Transportprotokollen für die Applikationsschnittstelle *Security-Layer*. Die nachfolgende Tabelle gibt für jedes dieser Transportprotokolle an, ob es in einer minimalen Umsetzung der *Bürgerkarten-Umgebung* enthalten sein muss (Anforderungs-Level ERFORDERLICH oder EMPFOHLEN) oder nicht (Anforderungs-Level OPTIONAL).

Transportprotokoll	Kontext	
	lokale BKU[(1)]	serverbasierte BKU[(1)]
TCP/IP	EMPFOHLEN	OPTIONAL
SSL/TLS	OPTIONAL	OPTIONAL
HTTP	ERFORDERLICH	OPTIONAL
HTTPS	ERFORDERLICH	ERFORDERLICH

(1) Für die Unterscheidung zwischen lokaler und serverbasierter *Bürgerkarten-Umgebung* siehe Bürgerkarten-Umgebung.

4. Profil für CMS-Signaturen

4.1. Signaturerstellung

Dieser Abschnitt spezifiziert ein Profil von [CMS], das von einer *Bürgerkarten-Umgebung* im Kontext des Befehls *CreateCMSSignature* verwendet werden MUSS.

4.1.1. Digest-Algorithmen

Wird in einer XML-Signatur der Signaturschlüssel der Keybox `SecureSignatureKeypair` verwendet (siehe Abschnitt 2.1, „Keybox für elektronische Signaturen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen), so MUSS zur Berechnung des *Message Digests* nach [CMS],

Abschnitt 5.4 ein nach [SigV] idgF. zulässiger Algorithmus verwendet werden.

> **Anmerkung**
>
> Da durch [SigV] für qualifizierte Signaturen kollisionsresistente Hash-Funktionen gefordert sind und aufgrund der Fortschritte in der Kryptoanalyse das Finden von Kollisionen für den Algorithmus SHA-1 erwartet wird, wird die Verwendung des Algorithmus SHA-1 nicht mehr empfohlen. Daher wird an dieser Stelle die Verwendung der Algorithmen SHA-256 bzw. RIPEMD160[1] zur Berechnung des *Message Digest* empfohlen.

4.1.2. Signaturalgorithmen

Der Algorithmus zur Berechnung des Signaturwerts nach [CMS], Abschnitt 5.5 ist abhängig vom verwendeten Signaturschlüssel der *Bürgerkarten-Umgebung*. Wird in einer CMS-Signatur der Signaturschlüssel der Keybox `SecureSignatureKeypair` verwendet (siehe Abschnitt 2.1, „Keybox für elektronische Signaturen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen), so MUSS zur Berechnung des Signaturwerts ein nach [SigV] idgF. zulässiger Algorithmus verwendet werden.

> **Anmerkung**
>
> Da durch [SigV] für qualifizierte Signaturen kollisionsresistente Hash-Funktionen gefordert sind und aufgrund der Fortschritte in der Kryptoanalyse das Finden von Kollisionen für den Algorithmus SHA-1 erwartet wird, wird die Verwendung von Algorithmen, die als Komponente den Algorithmus SHA-1 verwenden nicht mehr empfohlen. Daher wird an dieser Stelle die Verwendung von Algorithmen, die als Komponente, falls technisch möglich, den Algorithmus SHA-256, sonst, den Algorithmus RIPEMD160 verwenden, empfohlen.
>
> Für RSA-Schlüssel wird hier daher, falls technisch möglich, der Algorithmus RSA-SHA256, sonst, der Algorithmus RSA-RIPEMD160 und für ECDSA-Schlüssel, falls technisch möglich, der Algorithmus ECDSA-SHA256, sonst der Algorithmus ECDSA-RIPEMD160[2] empfohlen.

4.1.3. Schlüsselinformationen

In einer CMS-Signatur können Informationen zum Auffinden des Prüfschlüssels in Form einer Sammlung von X509-Zertifikaten im Feld `certificates` (vgl. [CMS], Abschnitt 5.1) angegeben werden. Die *Bürgerkarten-Umgebung* MUSS in diese Sammlung jedenfalls das Signatorzertifikat aufnehmen. Weiters wird EMPFOHLEN, die gesamte Kette an Zertifikaten, die zur Prüfung der Signatur benötigt wird (Signatorzertifikat bis zu einer vertrauenswürdigen Wurzelinstanz), ebenfalls in diese Sammlung aufzunehmen.

4.1.4. Auflösung von Referenzen

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Protokolle bei der Auflösung von Referenzen im Kontext des Befehls *CreateCMSSignature*.

Protokoll	**Kontext**
	`sl:DataObject/sl:Content/@Reference`
`http`	ERFORDERLICH
`https`	ERFORDERLICH
`formdata` (1)	ERFORDERLICH

(1) Vergleiche Abschnitt 3.2.1.2, „Referenzieren von Formularfeldern“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer.

4.2. Signaturprüfung

Dieser Abschnitt spezifiziert ein Profil von [CMS], das von einer *Bürgerkarten-Umgebung* im Kontext des Befehls *VerifyCMSSignature* mindestens beherrscht werden MUSS.

4.2.1. Digest-Algorithmen

Die *Bürgerkarten-Umgebung* MUSS alle Algorithmen zur Berechnung des *Message Digest* die im Rahmen der CMS-Signaturerstellung unterstützt werden auch im Rahmen der Signaturprüfung nach [CMS], Abschnitt 5.6 unterstützten.

> **Anmerkung**
>
> Um eine möglichst große Interoperabilität zwischen verschiedenen Umsetzungen einer *Bürgerkarten-Umgebung* zu erreichen, wird an dieser Stelle empfohlen, zumindest die in der Anmerkung zu Abschnitt 4.1.1, „Digest-Algorithmen“ angeführten Algorithmen (SHA-256, RIPEMD-160) und den Algorithmus SHA-1 zu unterstützten.

4.2.2. Signaturalgorithmen

Die *Bürgerkarten-Umgebung* MUSS alle Algorithmen zur Berechnung des Signaturwerts die im Rahmen der CMS-Signaturerstellung unterstützt werden auch im Rahmen der Signaturprüfung nach [CMS], Abschnitt 5.6 unterstützten.

> **Anmerkung**
>
> Um eine möglichst große Interoperabilität zwischen verschiedenen Umsetzungen einer *Bürgerkarten-Umgebung* zu erreichen, wird an dieser Stelle empfohlen, zumindest die in der Anmerkung zu Abschnitt 4.1.2, „Signaturalgorithmen“ angeführten Algorithmen (RSA-SHA256, ECDSA-SHA256, RSA-RIPEMD160, ECDSA-RIPEMD160) und die Algorithmen RSA bzw. DSA nach [CMS-Alg], 3.2 bzw. 3.1 und ECDSA nach [ECDSA-CMS], 2.2.1 zu unterstützten.

4.2.3. Schlüsselinformationen

In einer CMS-Signatur können Informationen zum Auffinden des Prüfschlüssels in Form einer Sammlung von X509-Zertifikaten im Feld `certificates` (vgl. [CMS], Abschnitt 5.1) angegeben sein. Die *Bürgerkarten-Umgebung* DARF diese Informationen zum Auffinden des Prüfschlüssels sowie zur Bildung der Zertifikatskette hin zu einer vertrauenswürdigen Wurzel verwenden.

Weiters können in einer CMS-Signatur Widerrufsinformationen in Form einer Sammlung von X509-Widerrufslisten im Feld `crls` (vgl. [CMS], Abschnitt 5.1) angegeben sein. Die *Bürgerkarten-Umgebung* DARF diese Informationen zur Feststellung des Status eines Zertifikats im Rahmen der Validierung einer Zertifikatskette verwenden.

4.2.4. Auflösung von Referenzen

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Protokolle bei der Auflösung von Referenzen im Kontext des Befehls *VerifyCMSSignature*.

Protokoll	**Kontext**
	`sl:DataObject/sl:Content/@Reference`
`http`	ERFORDERLICH
`https`	ERFORDERLICH
`formdata` (1)	ERFORDERLICH

(1) Vergleiche Abschnitt 3.2.1.2, „Referenzieren von Formularfeldern“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer.

5. Profil für XML-Signaturen

5.1. Signaturerstellung

Dieser Abschnitt spezifiziert ein Profil von [XMLDSIG], das von einer *Bürgerkarten-Umgebung* im Kontext des Befehls *CreateXMLSignature* verwendet werden MUSS.

5.1.1. Digest-Algorithmen

Wird in einer XML-Signatur der Signaturschlüssel der Keybox `SecureSignatureKeypair` verwendet (siehe Abschnitt 2.1, „Keybox für elektronische Signaturen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen), so MUSS zur Berechnung von *Message Digests* ein nach [SigV] idgF. zulässiger Algorithmus verwendet werden.

Anmerkung

Da durch [SigV] für qualifizierte Signaturen kollisionsresistente Hash-Funktionen gefordert sind und aufgrund der Fortschritte in der Kryptoanalyse das Finden von Kollisionen für den Algorithmus SHA-1 erwartet wird, wird die Verwendung des Algorithmus SHA-1 nach [XMLDSIG], Abschnitt 6.2.1 nicht mehr empfohlen. Daher wird an dieser Stelle die Verwendung der Algorithmen SHA-256 bzw. RIPEMD160 nach [XMLEnc], Abschnitt 5.7.2 bzw. 5.7.4 zur Berechnung von *Message Digests* empfohlen.

5.1.2. Signaturalgorithmen

Der Algorithmus zur Berechnung des Signaturwerts nach [XMLDSIG], Abschnitt 3.1.2 ist abhängig vom verwendeten Signaturschlüssel der *Bürgerkarten-Umgebung*. Wird in einer XML-Signatur der Signaturschlüssel der Keybox `SecureSignatureKeypair` verwendet (siehe Abschnitt 2.1, „Keybox für elektronische Signaturen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen), so MUSS zur Berechnung des Signaturwerts ein nach [SigV] idgF. zulässiger Algorithmus verwendet werden.

Anmerkung

Da durch [SigV] für qualifizierte Signaturen kollisionsresistente Hash-Funktionen gefordert sind und aufgrund der Fortschritte in der Kryptoanalyse das Finden von Kollisionen für den Algorithmus SHA-1 erwartet wird, wird die Verwendung von Algorithmen, die als Komponente den Algorithmus SHA-1 verwenden nicht mehr empfohlen. Daher wird an dieser Stelle die Verwendung von Algorithmen, die als Komponente, falls technisch möglich, den Algorithmus SHA-256, sonst, den Algorithmus RIPEMD160 verwenden, empfohlen.

Für RSA-Schlüssel wird hier daher, falls technisch möglich, der Algorithmus RSA-SHA256, nach [XMLDSIG-URI], Abschnitt 2.3.1, sonst, der Algorithmus RSA-RIPEMD160 nach [XMLDSIG-URI], Abschnitt 2.3.5, für ECDSA-Schlüssel, falls technisch möglich, der Algorithmus ECDSA-SHA256, nach [XMLDSIG-URI], Abschnitt 2.3.6, sonst der Algorithmus ECDSA-RIPEMD160[3] empfohlen.

5.1.3. Kanonisierungsalgorithmen

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Unterstützung von Kanonisierungsalgorithmen im Kontext des Befehls *CreateXMLSignature*.

Bezeichnung	**URI**	**normative Referenz**	**Anforderung**
C14N	`http://www.w3.org/TR/2001/REC-xml-c14n-20010315`	[XMLDSIG], 6.5.1	ERFORDERLICH
C14N with comments	`http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments`	[XMLDSIG], 6.5.1	ERFORDERLICH
EC14N	`http://www.w3.org/2001/10/xml-exc-c14n#`	[EC14N], 4	ERFORDERLICH
EC14N with comments	`http://www.w3.org/2001/10/xml-exc-c14n#WithComments`	[EC14N], 4	ERFORDERLICH

5.1.4. Transformationsalgorithmen

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Unterstützung von Transformationsalgorithmen im Kontext des Befehls *CreateXMLSignature*.

Bezeichnung	**URI**	**normative Referenz**	**Anforderung**
C14N	`http://www.w3.org/TR/2001/REC-xml-c14n-20010315`	[XMLDSIG], 6.5.1	ERFORDERLICH
C14N with comments	`http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments`	[XMLDSIG], 6.5.1	ERFORDERLICH
EC14N	`http://www.w3.org/2001/10/xml-exc-c14n#`	[EC14N], 4	ERFORDERLICH
EC14N with comments	`http://www.w3.org/2001/10/xml-exc-c14n#WithComments`	[EC14N], 4	ERFORDERLICH
Base64 Decoder	`http://www.w3.org/2000/09/xmldsig#base64`	[XMLDSIG], 6.6.2	ERFORDERLICH
XPath Filter 1	`http://www.w3.org/TR/1999/REC-xpath-19991116`	[XMLDSIG], 6.6.3	ERFORDERLICH
XPath Filer 2	`http://www.w3.org/2002/06/xmldsig-filter2`	[XPF2]	ERFORDERLICH
Enveloped Signature	`http://www.w3.org/2000/09/xmldsig#enveloped-signature`	[XMLDSIG], 6.6.4	ERFORDERLICH
XSLT	`http://www.w3.org/TR/1999/REC-xslt-19991116`	[XMLDSIG], 6.6.5	ERFORDERLICH
Binary Mode Decryption	`http://www.w3.org/2002/07/decrypt#Binary`	[XMLDecTF], 4	OPTIONAL
XML Mode Decryption(1)	`http://www.w3.org/2002/07/decrypt#XML`	[XMLDecTF], 3	OPTIONAL

(1) Bei der Verwendung einer *XML Mode Decryption* Transformation im Zuge der Erstellung einer XML-Signatur MUSS die *Bürgerkarten-Umgebung* den Mechanismus aus [XMLDecTF], Abschnitt 3.2 verwenden.

5.1.5. Schlüsselinformationen

In einer XML-Signatur können Informationen zum Auffinden des Prüfschlüssels im XML-Element `dsig:KeyInfo` (vgl. [XMLDSIG], Abschnitt 4.4) angegeben werden. Die *Bürgerkarten-Umgebung* MUSS in dieses XML-Element jedenfalls das Signatorzertifikat aufnehmen. Weiters wird EMPFOHLEN, die gesamte Kette an Zertifikaten, die zur Prüfung der Signatur benötigt wird (Signatorzertifikat bis zu einer vertrauenswürdigen Wurzelinstanz), ebenfalls in dieses XML-Element aufzunehmen.

5.1.6. Auflösung von Referenzen

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Protokolle bei der Auflösung von Referenzen im Kontext des Befehls *CreateXMLSignature*.

Protokoll		**http**	**https**	**formdata**	**ID-Referenz**	**XPointer-Referenz[2]**
Kontext	**`sl:DataObjectInfo/ sl:DataObject/ @Reference`**	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH	EMPFOHLEN
	`sl:DataObjectInfo/ sl:DataObject/ sl:LocRefContent`	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH	n.a.	n.a.
	Importierte Stylesheets[1]	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH	n.a.	n.a.
	`sl:DataObjectInfo/ sl:Supplement/ sl:Content/ sl:LocRefContent`	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH	n.a.	n.a.
	`sl:SignatureInfo/ sl:SignatureEnvironment/ @Reference`	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH	n.a.	n.a.
	`sl:SignatureInfo/ sl:Supplement/ sl:Content/ sl:LocRefContent`	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH	n.a.	n.a.

[1] Stylesheet-Imports, die in einer XSLT-Transformation zu einem Datenobjekt angegeben sind.

[2] *XPointer* nach [XPointer], die `xmlns` und `element` *XPointer Parts* enthalten.

5.2. Signaturprüfung

Dieser Abschnitt spezifiziert ein Profil von [XMLDSIG], das von einer *Bürgerkarten-Umgebung* im Kontext des Befehls *VerifyXMLSignature* mindestens beherrscht werden MUSS.

5.2.1. Digest-Algorithmen

Die *Bürgerkarten-Umgebung* MUSS alle Algorithmen zur Berechnung des *Message Digest* die im Rahmen der XML-Signaturerstellung unterstützt werden auch im Rahmen der Signaturprüfung nach [XMLDSIG], Abschnitt 3.2 unterstützten.

> **Anmerkung**
>
> Um eine möglichst große Interoperabilität zwischen verschiedenen Umsetzungen einer *Bürgerkarten-Umgebung* zu erreichen, wird an dieser Stelle empfohlen, zumindest die in der Anmerkung zu Abschnitt 5.1.1, „Digest-Algorithmen“ angeführten Algorithmen (SHA-256, RIPEMD160) und den Algorithmus SHA-1 nach [XMLDSIG], Abschnitt 3.2 zu unterstützten.

5.2.2. Signaturalgorithmen

Die *Bürgerkarten-Umgebung* MUSS alle Algorithmen zur Berechnung des Signaturwerts die im Rahmen der XML-Signaturerstellung unterstützt werden auch im Rahmen der Signaturprüfung nach [XMLDSIG], Abschnitt 3.2 unterstützten.

> **Anmerkung**
>
> Um eine möglichst große Interoperabilität zwischen verschiedenen Umsetzungen einer *Bürgerkarten-Umgebung* zu erreichen, wird an dieser Stelle empfohlen, zumindest die in der Anmerkung zu Abschnitt 5.1.2, „Signaturalgorithmen“ angeführten Algorithmen (RSA-SHA256, ECDSA-SHA256, RSA-RIPEMD160, ECDSA-RIPEMD160) und die Algorithmen RSA bzw. DSA nach [XMLDSIG], 6.4.2 bzw. 6.4.1 und ECDSA nach [XMLDSIG-URI], 2.3.6 zu unterstützten.

5.2.3. Kanonisierungsalgorithmen

Für die Tabelle der Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Unterstützung von Kanonisierungsalgorithmen im Rahmen der Signaturprüfung nach [XMLDSIG], Abschnitt 3.2 siehe Abschnitt 5.1.3, „Kanonisierungsalgorithmen“.

5.2.4. Transformationsalgorithmen

Für die Tabelle der Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Unterstützung von Transformationsalgorithmen im Rahmen der Signaturprüfung nach [XMLDSIG], Abschnitt 3.2 siehe Abschnitt 5.1.4, „Transformationsalgorithmen“.

Für diese beiden Transformationen Binary Mode Decryption und XML Mode Decryption gelten sinngemäß die Vorgaben aus Abschnitt 5.2, „Entschlüsselung eines XML-Dokuments“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer sowie jene aus diesem Dokument, Abschnitt 7.2, „Entschlüsselung“.

5.2.5. Schlüsselinformationen

In einer XML-Signatur können Informationen zum Auffinden des Prüfschlüssels im XML-Element `dsig:KeyInfo` (vgl. [XMLDSIG], Abschnitt 4.4) angegeben sein. Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Unterstützung von XML-Elementen, die darin vorkommen können. Alle nicht explizit in der Tabelle erwähnten XML-Elemente DÜRFEN von der *Bürgerkarten-Umgebung* ausgewertet werden.

Kindelement	**Anforderung**	**Anmerkung**
`dsig:RSAKeyValue`	EMPFOHLEN	-
`dsig:DSAKeyValue`	EMPFOHLEN	-
`dsm:ECDSAKeyValue`	EMPFOHLEN	-
`dsig:X509IssuerSerial`	EMPFOHLEN	Dieses XML-Element bezeichnet auf eindeutige Weise ein Zertifikat. Damit kann die *Bürgerkarten-Umgebung* ggf. ein Zertifikat aus dem eigenen Cache zuordnen.

`dsig:X509Certificate`	ERFORDERLICH	-
`dsig:X509CRL`	ERFORDERLICH	Eine mit diesem XML-Element kodierte Widerrufsliste muss zwar von der *Bürgerkarten-Umgebung* verstanden, aber nicht notwendigerweise verwendet werden (z.B. weil eine aktuellere Widerrufsliste von einem Verteilungspunkt abgerufen werden kann).
`dsig:RetrievalMethod` mit Verweis auf `dsig:X509Data`	ERFORDERLICH	Für die XML-Kindelemente des `dsig:X509Data`, auf das verwiesen wird, gelten wiederum die Anforderungslevels dieser Tabelle.
`dsig:RetrievalMethod` mit direktem Verweis auf ein X509- Zertifikat	ERFORDERLICH	-

5.2.6. Auflösung von Referenzen

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Protokolle bei der Auflösung von Referenzen im Kontext des Befehls *VerifyXMLSignature*.

Protokoll		**http**	**https**	**ldap**	**formdata**	**ID-Referenz**	**XPointer-Referenz**(2)
Kontext	**`sl:SignatureInfo/ sl:SignatureEnvironment/ @Reference`**	ERFORDERLICH	ERFORDERLICH	OPTIONAL	ERFORDERLICH	n.a.	n.a.
	`sl:Supplement/ sl:Content/ sl:LocRefContent`	ERFORDERLICH	ERFORDERLICH	OPTIONAL	ERFORDERLICH	n.a.	n.a.
	Importierte Stylesheets(1)	ERFORDERLICH	ERFORDERLICH	OPTIONAL	ERFORDERLICH	n.a.	n.a.
	`dsig:Signature/ dsig:SignedInfo/ dsig:Reference/ @URI`	ERFORDERLICH	ERFORDERLICH	OPTIONAL	ERFORDERLICH	ERFORDERLICH	EMPFOHLEN
	`dsig:Manifest/ dsig:Reference/ @URI`	ERFORDERLICH	ERFORDERLICH	OPTIONAL	ERFORDERLICH	ERFORDERLICH	EMPFOHLEN
	`dsig:Signature/ dsig:KeyInfo/ dsig:RetrievalMethod/ @URI`	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH	EMPFOHLEN

(1) Stylesheet-Imports, die in einer XSLT-Transformation der XML-Signatur enthalten sind.

(2) *XPointer* nach [XPointerFW], die `xmlns` (vgl. [XPointerNS]) und `element` (vgl. [XPointerEL]) *XPointer Parts* enthalten.

6. Profil für CMS-Verschlüsselung

6.1. Verschlüsselung

Dieser Abschnitt spezifiziert ein Profil von [CMS], das von einer *Bürgerkarten-Umgebung* im Kontext des Befehls *EncryptCMS* verwendet werden MUSS.

6.1.1. Algorithmen

Für erzeugte verschlüsselte CMS-Nachrichten MUSS eine *Bürgerkarten-Umgebung* für den Schlüsseltransport bzw. für die Datenverschlüsselung einen der in den Abschnitt 6.2.1.1, „Schlüsseltransport“ bzw. Abschnitt 6.2.1.2, „Datenverschlüsselung“ genannten Algorithmen verwenden.

Angesichts der weiten Verbreitung wird derzeit EMPFOHLEN, für den Schlüsseltransport den Algorithmus *PKCS #1 v1.5*, sowie für die Datenverschlüsselung den Algorithmus *Triple DES CBC* zu verwenden.

6.1.2. Auflösung von Referenzen

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Protokolle bei der Auflösung von Referenzen.

Protokoll	**Kontext**
	`sl:ToBeEncrypted/@Reference`
`http`	ERFORDERLICH
`https`	ERFORDERLICH
`formdata` (1)	ERFORDERLICH

(1) Vergleiche Abschnitt 3.2.1.2, „Referenzieren von Formularfeldern“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer

6.2. Entschlüsselung

Dieser Abschnitt spezifiziert ein Profil von [CMS], das von einer *Bürgerkarten-Umgebung* im Kontext des Befehls *DecryptCMS* mindestens beherrscht werden MUSS.

6.2.1. Algorithmen

Dieser Abschnitt spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Unterstützung von Algorithmen in der zu entschlüsselnden CMS-Nachricht. Prinzipiell muss eine *Bürgerkarten-Umgebung* hinsichtlich der Verschlüsselung des Datenverschlüsselungsschlüssels nur den Schlüsseltransport zu unterstützen. Deshalb werden in diesem Abschnitt auch keine Angaben zu anderen Techniken des Schlüsselmanagements gemacht.

6.2.1.1. Schlüsseltransport

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Unterstützung von Algorithmen zum Schlüsseltransport (Feld *ktri* der CMS-Nachricht).

Bezeichnung	**OID**	**normative Referenz**	**Anforderung**

PKCS #1 v1.5	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1 }	[CMS-Alg], 4.2.1	ERFORDERLICH
RSAES-OAEP	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 7 }	[CMS-RSAES-OAEP]	EMPFOHLEN

6.2.1.2. Datenverschlüsselung

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Unterstützung von Algorithmen zur Datenverschlüsselung (Feld *contentEncryptionAlgorithm* der CMS-Nachricht).

Bezeichnung	OID	normative Referenz	Anforderung
Triple-DES CBC	{ des-ede3-cbc OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) 7 }	[CMS-Alg], 5.1	ERFORDERLICH
AES CBC 128 Bit	{ joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor (3) nistAlgorithms(4) 1 2 }	[CMS-AES]	EMPFOHLEN
AES CBC 256 Bit	{ joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor (3) nistAlgorithms(4) 1 42 }	[CMS-AES]	EMPFOHLEN

6.2.2. Auflösung von Referenzen

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Protokolle bei der Auflösung von Referenzen.

Protokoll	Kontext
	sl:EncryptedContent/sl:Content/@Reference
http	ERFORDERLICH
https	ERFORDERLICH
formdata (1)	ERFORDERLICH

(1) Vergleiche Abschnitt 3.2.1.2, „Referenzieren von Formularfeldern“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer.

7. Profil für XML-Verschlüsselung

7.1. Verschlüsselung

Dieser Abschnitt spezifiziert ein Profil von [XMLEnc], das von einer *Bürgerkarten-Umgebung* im Kontext des Befehls *EncryptXML* verwendet werden MUSS.

7.1.1. Algorithmen

Für erzeugte verschlüsselte XML-Nachrichten MUSS die *Bürgerkarten-Umgebung* für die Datenverschlüsselung einen der im Abschnitt 7.2.1.1, „Algorithmen für xenc:EncryptedData“ genannten Algorithmen für Blockverschlüsselung, für den Schlüsseltransport einen der in Abschnitt 7.2.1.2, „Algorithmen für xenc:EncryptedKey“ genannten Algorithmen bzw. für die Schlüsselvereinbarung einen der in Abschnitt 7.2.1.3, „Algorithmen für xenc:KeyAgreement“ genannten Algorithmen verwenden.

Angesichts der weiten Verbreitung wird derzeit EMPFOHLEN, für den Schlüsseltransport den Algorithmus *RSA Version 1.5*, sowie für die Datenverschlüsselung den Algorithmus *AES CBC 128 Bit* zu verwenden.

7.1.2. Auflösung von Referenzen

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Protokolle bei der Auflösung von Referenzen.

Protokoll	Kontext		
	sl:ToBeEncrypted/ sl:New/ sl:LocRefContent	sl:EncryptionInfo/ sl:EncryptionEnvironment/ @Reference	sl:EncryptionInfo/ sl:Supplement/ sl:Content/ sl:LocRefContent
http	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH
https	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH
formdata (1)	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH

(1) Vergleiche Abschnitt 3.2.1.2, „Referenzieren von Formularfeldern“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer

7.2. Entschlüsselung

Dieser Abschnitt spezifiziert ein Profil von [XMLEnc], das von einer *Bürgerkarten-Umgebung* im Kontext des Befehls *DecryptXML* mindestens beherrscht werden MUSS.

7.2.1. Algorithmen

7.2.1.1. Algorithmen für xenc:EncryptedData

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend Algorithmen in xenc:EncryptedData/xenc:EncryptionMethod/@Algorithm.

Bezeichnung	URI	normative Referenz	Anforderung
Blockverschlüsselung			
Triple DES	http://www.w3.org/2001/04/xmlenc#tripledes-cbc	[XMLEnc], 5.2.1	ERFORDERLICH
AES CBC 128 Bit	http://www.w3.org/2001/04/xmlenc#aes128-cbc	[XMLEnc], 5.2.2	ERFORDERLICH
AES CBC 256 Bit	http://www.w3.org/2001/04/xmlenc#aes256-cbc	[XMLEnc], 5.2.2	ERFORDERLICH
Schlüsseltransport (1)			
RSA Version 1.5	http://www.w3.org/2001/04/xmlenc#rsa-1_5	[XMLEnc], 5.4.1	ERFORDERLICH

RSA-OAEP	`http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p`	[XMLEnc], 5.4.2	ERFORDERLICH

(1)Siehe Hinweis in [XMLEnc], Abschnitt 5.4.

7.2.1.2. Algorithmen für `xenc:EncryptedKey`

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend Algorithmen in `xenc:EncryptedKey/xenc:EncryptionMethod/@Algorithm`.

Bezeichnung	**URI**	**normative Referenz**	**Anforderung**
Schlüsseltransport			
RSA Version 1.5	`http://www.w3.org/2001/04/xmlenc#rsa-1_5`	[XMLEnc], 5.4.1	ERFORDERLICH
RSA-OAEP	`http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p`	[XMLEnc], 5.4.2	ERFORDERLICH
Schlüsselverschlüsselung (Symetric Key Wrap)			
CMS Triple DES Key Wrap	`http://www.w3.org/2001/04/xmlenc#kw-tripledes`	[XMLEnc], 5.6.2	EMPFOHLEN
AES KeyWrap 128 Bit	`http://www.w3.org/2001/04/xmlenc#kw-aes128`	[XMLEnc], 5.6.3	EMPFOHLEN
AES KeyWrap 256 Bit	`http://www.w3.org/2001/04/xmlenc#kw-aes256`	[XMLEnc], 5.6.3	EMPFOHLEN
Blockverschlüsselung			
Triple DES	`http://www.w3.org/2001/04/xmlenc#tripledes-cbc`	[XMLEnc], 5.2.1	ERFORDERLICH
AES CBC 128 Bit	`http://www.w3.org/2001/04/xmlenc#aes128-cbc`	[XMLEnc], 5.2.2	ERFORDERLICH
AES CBC 256 Bit	`http://www.w3.org/2001/04/xmlenc#aes256-cbc`	[XMLEnc], 5.2.2	ERFORDERLICH

7.2.1.3. Algorithmen für `xenc:KeyAgreement`

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend Algorithmen in `xenc:EncryptedKey/xenc:KeyInfo/xenc:KeyAgreement/@Algorithm`.

Bezeichnung	**URI**	**normative Referenz**	**Anforderung**
Schlüsselvereinbarung			
Diffie-Hellman Key Agreement	`http://www.w3.org/2001/04/xmlenc#dh`	[XMLEnc], 5.5.2	EMPFOHLEN

7.2.2. Schlüsselhinweise

7.2.2.1. Schlüsselhinweise in `xenc:EncryptedData`

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend Schlüsselhinweise in `xenc:EncryptedData/dsig:KeyInfo`.

Elementname	**Anmerkung**	**normative Referenz**	**Anforderung**
`dsig:KeyName`	MUSS den *KeyboxIdentifier* eines in der *Bürgerkarten-Umgebung* vorhandenen Schlüsselpaares enthalten, das für Verschlüsselung geeignet ist.	[XMLDSIG], 4.4.1	ERFORDERLICH
`dsig:KeyValue`	MUSS den öffentlichen Schlüssel eines in der *Bürgerkarten-Umgebung* vorhandenen Schlüsselpaares enthalten, das für Verschlüsselung geeignet ist.	[XMLDSIG], 4.4.2	ERFORDERLICH
`dsig:X509Data`	MUSS genau ein Element `dsig:X509Certificate` mit dem Zertifikat für den öffentlichen Schlüssel eines in der *Bürgerkarten-Umgebung* vorhandenen Schlüsselpaares enthalten.	[XMLDSIG], 4.4.4	ERFORDERLICH
`xenc:EncryptedKey`		[XMLEnc], 3.5.1	ERFORDERLICH
`dsig:RetrievalInfo`	MUSS auf ein Element `xenc:EncryptedKey` im gleichen XML-Dokument verweisen. Transformationen müssen von der *Bürgerkarten-Umgebung* nicht unterstützt werden.	[XMLEnc], 3.5.2	ERFORDERLICH

7.2.2.2. Schlüsselhinweise in `xenc:EncryptedKey`

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend Schlüsselhinweise in `xenc:EncryptedKey/dsig:KeyInfo`.

Elementname	**Anmerkung**	**normative Referenz**	**Anforderung**
`dsig:KeyName`	MUSS den *KeyboxIdentifier* eines in der *Bürgerkarten-Umgebung* vorhandenen Schlüsselpaares enthalten, das für Verschlüsselung geeignet ist.	[XMLDSIG], 4.4.1	ERFORDERLICH
`dsig:KeyValue`	MUSS den öffentlichen Schlüssel eines in der *Bürgerkarten-Umgebung* vorhandenen Schlüsselpaares enthalten, das für Verschlüsselung geeignet ist.	[XMLDSIG], 4.4.2	ERFORDERLICH
`dsig:X509Data`	MUSS genau ein Element `dsig:X509Certificate` mit dem Zertifikat für den öffentlichen Schlüssel eines in der *Bürgerkarten-Umgebung* vorhandenen Schlüsselpaares enthalten.	[XMLDSIG], 4.4.4	ERFORDERLICH
`xenc:AgreementMethod`	• Das Attribut `xenc:AgreementMethod/@Algorithm` MUSS die URI eines unter Abschnitt 7.2.1.3, „Algorithmen für `xenc:KeyAgreement`“ angeführten Algorithmus zur Schlüsselvereinbarung enthalten. • Das Element `xenc:AgreementMethod/dsig:DigestMethod` MUSS einen unter Abschnitt 5.1.1, „Digest-Algorithmen“ angeführten Algorithmus referenzieren. • Das Element `xenc:AgreementMethod/xenc:RecipientKeyInfo` MUSS genau eines der Elemente `dsig:KeyName`, `dsig:KeyValue`, `dsig:X509Data` wie oben enthalten. • Das Element `xenc:AgreementMethod/xenc:OriginatorKeyInfo` MUSS genau ein Element `dsig:KeyValue` enthalten.	[XMLEnc], 5.5	OPTIONAL

7.2.3. Auflösung von Referenzen

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Protokolle bei der Auflösung von Referenzen in unterschiedlichen Kontexten.

Protokoll	**Kontext**			
	`dsig:RetrievalMethod`	`xenc:EncryptedData/ xenc:Value`	`xenc:EncryptedKey/ xenc:Value`	`sl:Supplement/ sl:Content/ sl:LocRefContent`
`http`	OPTIONAL	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH
`https`	OPTIONAL	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH
`formdata` (1)	OPTIONAL	ERFORDERLICH	ERFORDERLICH	ERFORDERLICH
relative URI (2)	optional(4)	erforderlich(4)	erforderlich(4)	darf nicht
ID-Referenz (3)	ERFORDERLICH	OPTIONAL	OPTIONAL	darf nicht

(1) Vergleiche Abschnitt 3.2.1.2, „Referenzieren von Formularfeldern“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer.

(2) Eine relative URI ist eine URI, die keinen Protokoll-Teil enthält (vgl. [URI], 3.1).

(3) Eine ID-Referenz ist eine *Same Document Reference* nach [URI], 4.2, die den Wert eines Attributs vom Typ *IDREF* nach [XML] enthält.

(4) Eine relative URI darf nur über ein für diese URI angegebenes *Supplement* aufgelöst werden; eine direkte Auflösung (z.B. über ein Absolut-Machen gegen das lokale Dateisystem) DARF NICHT erfolgen.

8. Profil für Hashwerte

8.1. Hashwert-Berechnung

Dieser Abschnitt spezifiziert ein Profil für den Befehl *CreateHash*, das von einer *Bürgerkarten-Umgebung* mindestens beherrscht werden MUSS.

8.1.1. Digest-Algorithmen

Die *Bürgerkarten-Umgebung* MUSS für die Berechnung des *Message Digest* im Rahmen der Hashwert-Berechnung alle Algorithmen unterstützen, die auch im Rahmen der XML-Signaturerzeugung unterstützt werden.

8.1.2. Auflösung von Referenzen

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Protokolle bei der Auflösung von Referenzen.

Protokoll	**Kontext**
	`sl:HashInfo/ sl:HashData/ sl:Content/ @Reference`
`http`	ERFORDERLICH
`https`	ERFORDERLICH
`formdata` (1)	ERFORDERLICH

(1) Vergleiche Abschnitt 3.2.1.2, „Referenzieren von Formularfeldern“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer

8.2. Hashwert-Verifikation

Dieser Abschnitt spezifiziert ein Profil für den Befehl *VerifyHash*, das von einer *Bürgerkarten-Umgebung* mindestens beherrscht werden MUSS.

8.2.1. Digest-Algorithmen

Die *Bürgerkarten-Umgebung* MUSS für die Berechnung des *Message Digest* im Rahmen der Hashwert-Verifikation alle Algorithmen unterstützen, die auch im Rahmen der XML-Signaturprüfung unterstützt werden.

8.2.2. Auflösung von Referenzen

Die nachfolgende Tabelle spezifiziert die Anforderungslevels an die *Bürgerkarten-Umgebung* betreffend die Protokolle bei der Auflösung von Referenzen.

Protokoll	**Kontext**
	`sl:HashInfo/ sl:HashData/ sl:Content/ @Reference`
`http`	ERFORDERLICH
`https`	ERFORDERLICH
`formdata` (1)	ERFORDERLICH

(1) Vergleiche Abschnitt 3.2.1.2, „Referenzieren von Formularfeldern“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer

9. Anzeigeformate und Zeichensätze

9.1. Formate für die Anzeige der Bürgerkarten-Umgebung

Für die Abarbeitung der Schnittstellenbefehle zur Signaturerstellung (*CreateCMSSignature*, *CreateXMLSignature*) sowie zur Signaturprüfung (*VerifyCMSSignature*, *VerifyXMLSignature*) benötigt die *Bürgerkarten-Umgebung* eine Anzeige, um dem Bürger die zu signierenden bzw. die signierten Daten darstellen zu können. Dieser Abschnitt legt fest, welche Anzeigeformate die Anzeige der *Bürgerkarten-Umgebung* jedenfalls unterstützen muss.

9.1.1. Text

Als einfaches Format MUSS die Anzeige der *Bürgerkarten-Umgebung* die Darstellung von Text beherrschen. Die nachfolgende Tabelle listet jene Unicode-Zeichen [Unicode] auf, die jedenfalls dargestellt werden können MÜSSEN.

Anmerkung: Es handelt sich dabei um jene Zeichen, die notwendig sind, um die Zeichensätze [ISO-8859-1], [ISO-8859-2], [ISO-8859-3], [ISO-8859-9], [ISO-8859-10] und [ISO-8859-15] mit Ausnahme der meisten dort enthaltenen Steuerzeichen darstellen zu können.

Code Chart	Zeichennummer(n)	Code Chart	Zeichennummer(n)
C0 Controls and Basic Latin	0x0009-0x000A	Latin Extended-A	0x014A-0x014D
C0 Controls and Basic Latin	0x000C-0x000D	Latin Extended-A	0x0150-0x0155
C0 Controls and Basic Latin	0x0020-0x007E	Latin Extended-A	0x0158-0x0173
C1 Controls and Latin-1 Supplement	0x00A1-0x00FF	Latin Extended-A	0x0178-0x017E
Latin Extended-A	0x0100-0x0114	Spacing Modifier Letters	0x02C7
Latin Extended-A	0x0116-0x012B	Spacing Modifier Letters	0x02D8-0x02D9
Latin Extended-A	0x012E-0x0131	Spacing Modifier Letters	0x02DB
Latin Extended-A	0x0134-0x013E	Spacing Modifier Letters	0x02DD
Latin Extended-A	0x0141-0x0148	General Punctuation	0x2015

Folgende Daten muss die *Bürgerkarten-Umgebung* versuchen, als Text zu interpretieren:

- Daten, die mit dem *MIME Media Type* [MIME-Media] `text/plain` gekennzeichnet sind;
- Daten, für die keine Informationen über den *MIME Media Type* vorliegen;
- Daten, die mit einem *der folgenden MIME Media Types* gekennzeichnet sind, wenn die *Bürgerkarten-Umgebung* für diesen *Media Type* keine eigene - über die in Abschnitt 9.1, „Formate für die Anzeige der Bürgerkarten-Umgebung“ beschriebenen Formate hinausgehende - Anzeigemöglichkeit bietet: `text/tab-separated-values`, `text/sgml`, `text/xml`, `application/sgml`, `application/xml`, `message/rfc822`.

Für die Darstellung von Text MUSS die *Bürgerkarten-Umgebung* eine Schriftart mit konstanter Breite für jedes Zeichen (*monospaced*) verwenden. Für das Zeichen `0x0009` (Tabulator) MUSS die *Bürgerkarten-Umgebung* eine Tabulatorbreite von acht Zeichen verwenden.

9.1.2. Standard-Anzeigeformat des Security-Layers

Als Format für die Darstellung von komplexeren Dokumenten MUSS die *Bürgerkarten-Umgebung* die Darstellung des Standard-Anzeigeformat beherrschen. Hinsichtlich der Zeichen, die die *Bürgerkarten-Umgebung* jedenfalls darstellen können muss, gelten die Vorgaben aus Abschnitt 9.1.1, „Text“.

Folgende Daten muss die *Bürgerkarten-Umgebung* versuchen, im Sinne dieses Formats zu interpretieren:

- Daten, die mit dem *MIME Media Type* [MIME-Media] `text/html` gekennzeichnet sind;
- Daten, die mit dem *MIME Media Type* [MIME-Media] `application/xhtml+xml` gekennzeichnet sind.

9.2. Zeichensätze für das Schnittstellenprotokoll

Beim Empfang von Anfrage-Befehlen über die Schnittstelle des *Security-Layer* MUSS die *Bürgerkarten-Umgebung* die drei Zeichensätze ISO-8859-1 [ISO-8859-1], ISO-8859-15 [ISO-8859-15] sowie UTF-8 [Unicode] unterstützen. Die *Applikation* MUSS den verwendeten Zeichensatz in der XML-Deklaration des XML-Dokuments angeben, welches den Anfrage-Befehl beinhaltet.

Glossar

Glossar

Applikation

Jenes Programm, das Anfragen an die *Bürgerkarten-Umgebung* über den *Security-Layer* richtet und die entsprechenden Antworten entgegennimmt und auswertet.

Benutzer-Schnittstelle

Jene Schnittstelle, über die der *Bürger* mit der *Bürgerkarten-Umgebung* kommuniziert. Über diese Schnittstelle wird einerseits die Benutzerinteraktion abgewickelt, die gegebenenfalls zur Abwicklung eines Befehls des *Security-Layers* notwendig ist (z.B. die Anzeige eines zu signierenden Dokuments beim Befehl zur Erzeugung einer XML-Signatur); andererseits kann der *Bürger* über diese Schnittstelle seine *Bürgerkarten-Umgebung* nach seinen persönlichen Bedürfnissen konfigurieren (z.B. kann er Einstellungen zum Zugriffsschutz auf seine Infoboxen verändern). Die Vorgaben an die *Benutzer-Schnittstelle* sind in Minimale Umsetzung des Security-Layers geregelt.

Bürger

Jene Person, die die Funktionen der *Bürgerkarten-Umgebung* für die sichere Abwicklung von E-Government oder E-Commerce verwenden möchte. Die Ansteuerung der *Bürgerkarten-Umgebung* erfolgt in der Regel nicht durch den *Bürger* selbst, sondern durch die *Applikation*, welche die E-Government oder E-Commerce Anwendung repräsentiert.

Bürgerkarte

Laut [E-GovG], §10 ZI 10 ist die *Bürgerkarte* „die unabhängig von der Umsetzung auf unterschiedlichen technischen Komponenten gebildete logische Einheit, die eine elektronische Signatur mit einer Personenbindung (§ 4 Abs. 2) und den zugehörigen Sicherheitsdaten und -funktionen sowie mit allenfalls vorhandenen Vollmachtsdaten verbindet“. Im Sinne der in den Spezifikationen zur österreichischen Bürgerkarte gebrauchten Terminologie ist die *Bürgerkarten-Umgebung* die Implementierung der logischen Einheit *Bürgerkarte*.

Bürgerkarten-Umgebung

Jenes Programm bzw. jener Dienst, der die Funktionalität der *Bürgerkarte* zur Verfügung stellt. Grundsätzlich vorstellbar ist die Ausführung als Programm, das lokal am Rechner des *Bürgers* läuft (*lokale Bürgerkarten-Umgebung*), oder als serverbasierter Dienst, der über das Internet angesprochen wird (*serverbasierte Bürgerkarten-Umgebung*). Die Interaktion mit diesem Programm bzw. Dienst wird über zwei Schnittstellen abgewickelt: Über die *Benutzer-Schnittstelle* sowie über den *Security-Layer*.

Hash-Eingangsdaten

Jene Daten, die für die Berechnung des Hash-Wertes für eine `dsig:Reference` verwendet werden. Sind für die `dsig:Reference` Transformationen angegeben, entsprechen diese Daten dem Ergebnis der letzten Transformation. Sind keine Transformationen spezifiziert, gleichen die Hash-Eingangsdaten den *Referenz-Eingangsdaten*.

Impliziter Transformationsparameter

Siehe Abschnitt 2.2.2.2, „Implizite Transformationsparameter“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

Referenz-Eingangsdaten

Jene Daten, die sich aus der Auflösung der im Attribut `URI` der `dsig:Reference` angegebenen URI ergeben. Sind für die `dsig:Reference` Transformationen angegeben, werden diese Daten als Eingangsdaten zur Berechnung der ersten Transformation verwendet. Sind keine Transformationen spezifiziert, gleichen die Referenz-Eingangsdaten den *Hash-Eingangsdaten*.

Security-Layer

Jene Schnittstelle, über die die *Applikation* mit der *Bürgerkarten-Umgebung* kommuniziert. Das genaue Protokoll, das über diese Schnittstelle gesprochen werden kann, wird in Applikationsschnittstelle Security-Layer spezifiziert. Die möglichen Bindungen dieses Protokolls an Transportschichten wie HTTP oder TCP wird in Transportprotokolle Security-Layer geregelt.

Signaturmanifest

Siehe Abschnitt 2.2.2.2, „Implizite Transformationsparameter“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer .

Referenzen

[CMS] BHously, R.: *RFC 3369: Cryptographic Message Syntax (CMS)* , IETF Request For Comment, August 2002

[CMS-AES] chaad, J.: *RFC 3565: Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)* . IETF Request For Comment, Juli 2003.

[CMS-Alg] Hously, R.: *RFC 3370: Cryptographic Message Syntax (CMS) Algorithms* . IETF Request For Comment, August 2002.

[CMS-RSAES-OAEP] Hously, R.: *RFC 3560: Use of the RSAES-OAEP Key Transport Algorithm in the Cryptographic Message Syntax (CMS)* . IETF Request For Comment, Juli 2003.

[CSS 2] Bert Bos, Håkon Wium Lie, Chris Lilley und Ian Jacobs: *Cascading Style Sheets, level 2* . W3C Recommendation, Mai 1998.

[EC14N] Boyer, John, Eastlake, Donald und Reagle, Joseph: *Exclusive XML Canonicalization. W3C Recommendation, Juli 2002* .

[ECDSA-CMS] Blake-Wilson, S., Brown, D., Lampert, D.: *RFC 3278: Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)* . IETF Request For Comment, April 2002.

[ECDSA-XML] Blake-Wilson, S., Karlinger, G. und Wang, Y.: *ECDSA with XML-Signature Syntax* . Internet-Draft, Jänner 2004.

[E-GovG] *BGBl. I Nr. 10/2004.*

[ESS-S/MIME] Hoffman, P.: *RFC 2634: Enhanced Security Services for S/MIME* , IETF Request For Comment, Juni 1999

[ETSICMS] European Telecommunications Standards Institute: *ETSI TS 101733: Electronic Signature Formats, v1.5.1* , Technical Specification, Dezember 2003

[ETSIQCert] European Telecommunications Standards Institute: *ETSI TS 101 862: Qualified certificate profile, v1.2.1* , Technical Specification, Juni 2001

[ETSIXML] European Telecommunications Standards Institute: *ETSI TS 101903: XML Advanced Electronic Signatures (XAdES), v1.2.2* , Technical Specification, April 2004

[GIF] *Graphics Interchange Format, Version 89a* . CompuServe Incorporated, Juli 1990.

[HTML4] Dave Ragget, Arnaud Le Hors und Ian Jacobs: *HTML 4.01 Specification* . W3C Recommendation, Dezember 1999.

[HTTP1.1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leech und T. Berners-Lee: *Hypertext Transfer Protocol -- HTTP/1.1*. IETF Request For Comment, Juni 1999.

[HTTPS] E. Rescorla *HTTP over TLS*. IETF Request For Comment, Mai 2000

[ISO-8859-1] *ISO/IEC 8859-1*:1998: Information technology -- 8-bit single-byte coded graphic character sets -- Part 1: Latin alphabet No. 1.

[ISO-8859-10] *ISO/IEC 8859-10*:1998: Information technology -- 8-bit single-byte coded graphic character sets -- Part 10: Latin alphabet No. 6.

[ISO-8859-15] *ISO/IEC 8859-15*:1999: Information technology -- 8-bit single-byte coded graphic character sets -- Part 15: Latin alphabet No. 9.

[ISO-8859-2] *ISO/IEC 8859-2*:1999: Information technology -- 8-bit single-byte coded graphic character sets -- Part 2: Latin alphabet No. 2.

[ISO-8859-3] *ISO/IEC 8859-3*:1999: Information technology -- 8-bit single-byte coded graphic character sets -- Part 3: Latin alphabet No. 3.

[ISO-8859-9] *ISO/IEC 8859-9*:1999: Information technology -- 8-bit single-byte coded graphic character sets -- Part 9: Latin alphabet No. 5.

[JPEG] Eric Hamilton: *JPEG File Interchange Format, Version 1.02* . C-Cube Microsystems, September 1992.

[KEYWORDS] Bradner, S.: *RFC 2119: Key words for use in RFCs to Indicate Requirement Levels* , IETF Request For Comment, März 1997

[MIME] Freed, N. und Borenstein, N.: *RFC 2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types* , IETF Request For Comment, November 1996

[PersBin] Hollosi, Arno und Karlinger, Gregor: *XML-Definition der Personenbindung* . Konvention zum E-Government Austria erarbeitet von der Stabsstelle IKT-Strategie des Bundes, Technik und Standards. Öffentlicher Entwurf, Version 1.2.2, 14. Februar 2005.

[PersonData] Naber, Larissa: *PersonData Struktur - XML Spezifikation* . Konvention zum E-Government Austria erarbeitet von der Arbeitsgruppe Kommunikationsarchitekturen. Öffentlicher Entwurf, Version 2.0.0, 14. Oktober 2004.

[PKCS#12] RSA Laboratories: *PKCS#12 v1.0: Personal Information Exchange Syntax* , Juni 1999.

[port-numbers] Internet Assigned Numbers Authority: *Port Numbers*

[QCert] Santesson, S. und Nystrom M.: *RFC 3739: Internet X.509 Public Key Infrastructure: Qualified Certificates Profile* , IETF Request For Comment, März 2004

[SigG] *BGBl I Nr. 190/1999* idF BGBl I Nr. 152/2001.

[SigV] *BGBl II Nr. 30/2000* idF BGBl II Nr. 527/2004.

[Stammzahl] Hollosi, Arno und Hörbe, Rainer: *Bildung von Stammzahl und bereichsspezifischem Personenkennzeichen (bPK)* . Konvention zum E-Government Austria erarbeitet von der Stabsstelle IKT-Strategie des Bundes, Technik und Standards sowie vom Bundesminsterium für Inneres. Öffentlicher Entwurf, Version 1.0, 2. Februar 2004.

[TLS] T. Dierks und C. Allen: *The TLS Protocol Version 1.0* . IETF Request For Comment, Januar 1999.

[Unicode] The Unicode Consortium. *The Unicode Standard, Version 4.0.0* , defined by: The Unicode Standard, Version 4.0 (Boston, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1).

[URI] Berners-Lee, T. , Fielding, R. und Masinter, L.: *RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax* , IETF Request For Comment, August 1998

[VerwEig] Hollosi, Arno: *X.509 Zertifikatserweiterungen für die Verwaltung* . Konvention zum E-Government Austria erarbeitet von der Stabsstelle IKT-Strategie des Bundes, Technik und Standards. Öffentlicher Entwurf, Version 1.0.3, 21. Februar 2005.

[X509] Polk, W., Ford, W., Solo, D.: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* . IETF Request For Comment, April 2002.

[XHTML 1.1] Murray Altheim, Frank Boumphrey, Sam Dooley, Shane McCarron, Sebastian Schnitzenbaumer und Ted Wugofski: *Modularization of XHTML* . W3C Recommendation, April 2001.

[XHTML MOD] Daniel Austin, Subramanian Peruvemba, Shane McCarron, Masayasu Ishikawa: *Modularization of XHTML in XML Schema* . W3C Working Draft, Oktober 2003.

[XML] Bray, Tim, Paoli, Jean, Sperberg-McQueen, C.M. und Maler, Eve: *Extensible Markup Language (XML) 1.0 (Second Edition)* , W3C Recommendation, Oktober 2000.

[XMLDecTF] Hughes, Merlin, Imamura, Takeshi und Maruyama, Hiroshi: *Decryption Transform for XML Signature* . W3C Recommendation, Dezember 2002.

[XMLDSIG] Eastlake, Donald, Reagle, Joseph und Solo, David: *XML-Signature Syntax and Processing* , W3C Recommendation, Februar 2002

[XMLDSIG-URI] Eastlake, Donald: *RFC 4051: Additional XML Security Uniform Resource Identifiers (URIs)* , IETF Request For Comments, April 2005

[XMLEnc] Eastlake, Donald und Reagle, Joseph: *XML Encryption Syntax and Processing* , W3C Recommendation, Dezember 2002

[XML-Schema] Thompson, Henry S., Beech, David, Maloney, Murray und Mendelson, Noah: *XML Schema Part 1: Structures* , W3C Recommendation, Mai 2001

[XMLTYPE] Murata, M., St.Laurent, S., und Kohn, D.: *RFC 3023: XML Media Types* , IETF Request For Comment, Jänner 2001.

[XPath] Clark, James und DeRose, Steven: *XML Path Language* , W3C Recommendation, November 1999

[XPF2] Boyer, John, Hughes, Merlin und Reagle, Joseph: *XML-Signature XPath Filter 2.0* . W3C Candidate Recommendation, Juli 2002.

[XPointer] Grosso, Paul, Maler, Eve, Marsh, Jonathan und Walsh, Norman: *XPointer Framework* . W3C Recommendation, März 2003.

[XSS-FAQ] Cgisecurity.com: *The Cross Site Scripting FAQ* .

A. Historie

Datum	Version	Änderungen
20.02.2008	1.2.1	• Zulässige Digest-Algorithmen in Abschnitt 4, Abschnitt 5 und Abschnitt 8 erweitert. • Profil für XML-Verschlüsselung in Abschnitt 7 um Schlüsselvereinbarung (key agreement) erweitert.
29.02.2004	1.2.0	• Erratum 3 laut Errata ausgebessert. • Schnittstellenbefehl *CreateSymmetricSecret* aus Liste der Schnittstellenbefehle entfernt. • Schnittstellenbefehle *EncryptCMS*, *EncryptXML*, *DecryptCMS*, *DecryptXML*, *CreateHash*, *VerifyHash*, *InfoboxCreate*, *InfoboxDelete*, *NullOperation* in die Liste der Schnittstellenbefehle hinzugefügt. • Abschnitte 4 über Profil für CMS-Signaturen hinzugefügt. • Ehemaligen Abschnitt 3 über XML-Signaturen in Abschnitt 5 neu organisert (Trennung in Erstellung und Prüfung). • Abschnitte 6 und 7 über Profil für CMS-Verschlüsselung und XML-Verschlüsselung hinzugefügt. • Abschnitt 8 über Profil für Hashwert-Berechnung und -Verifikation hinzugefügt. • Abschnitt 9 über Anzeigeformate und Zeichensätze hinzugefügt. • Ehemaligen Abschnitt 8 in Unterkapitel der Abschnitte 4 bis 7 eingegliedert.
31.08.2002	1.1.0	• Schnittstellenbefehle Signaturerstellung und Signaturprüfung nach CMS von erforderlich auf empfohlen geändert. • Schnittstellenbefehl zur Erzeugung eines Sitzungszertifikats entfernt. • Profil für XML-Signaturen (Prüfung einer XML-Signatur, Erstellung einer XML-Signatur, Aushandeln eines symmetrischen Geheimnisses) hinzugefügt. • Anforderungen an die Auflösung von URIs in Request-Befehlen und zu prüfenden XML-Signaturen hinzugefügt. • Abschnitt 1.1 Namenskonventionen hinzugefügt.

[1] Die Verwendung der Algorithmen SHA-256 und RIPEMD160 mit CMS ist noch nicht normativ definiert. http://www.ietf.org/internet-drafts/draft-ietf-smime-sha2-01.txt definiert die Verwendung der SHA2-Algorithmen mit CMS.

[2] Die Verwendung der Algorithmen RSA-SHA256, RSA-RIPEMD160, ECDSA-SHA256 und ECDSA-RIPEMD160 mit CMS ist noch nicht normativ definiert. http://www.ietf.org/internet-drafts/draft-ietf-smime-sha2-01.txt definiert die Verwendung der Algorithmen RSA-SHA256 und ECDSA-SHA256 mit CMS.

[3] Die Verwendung des Algorithmus ECDSA-RIPEMD160 mit XMLDSIG wurde bisher noch nicht normativ definiert. http://www.ietf.org/internet-drafts/draft-eastlake-additional-xmlsec-uris-00.txt definiert die Verwendung des Algorithmus ECDSA-RIPEMD160 und den URI http://www.w3.org/2007/05/xmldsig-more#ecdsa-ripemd160.