
Mehrfachsignaturen

Konzept – Version 1.0.1

Dipl.-Ing. Martin Centner

Zusammenfassung

Als Mehrfachsignaturen werden mehrere Signaturen über die selben Daten bezeichnet. Dabei kann zwischen voneinander unabhängigen Signaturen und Signaturen, die bereits existierende Signaturen über die selben Daten einschließen, unterschieden werden. Bei Signaturen die andere Signaturen einschließen, gibt es wiederum unterschiedliche Möglichkeiten zum Verketteten der Signaturen.

Das vorliegende Dokument zeigt Möglichkeiten zum Verketteten von *XML Digital Signatures* nach dem Konzept Bürgerkarte und diskutiert Punkte, die bei Signaturen über die selben Daten und beim Einbetten mehrerer Signaturen in ein Dokument zu beachten sind.

Inhaltsverzeichnis

1	Einleitung	2
2	Mehrfachsignaturen	2
3	Verketteten von Signaturen	3
4	Referenzieren der Daten	6
5	Einbetten mehrerer Signaturen in ein Dokument	7
6	Zusammenfassung	8
A	Beispiel für ein Reference-Element zur Verketteten von Signaturen	9

1 Einleitung

Wie bei handschriftlich unterschriebenen Dokumenten, können auch bei elektronisch signierten Dokumenten mehrere unterschiedliche Signaturen angebracht werden. Bei elektronisch signierten Dokumenten kann jedoch viel genauer spezifiziert werden, auf welche Teile des Dokuments sich die Signatur bezieht und in welcher Beziehung mehrere vorhandene Unterschriften zueinander stehen, d.h. über welche Daten die elektronische Unterschrift tatsächlich erstellt wird. So lässt sich z.B. eine bereits existierende elektronische Signatur in die Daten, über die die eigene Signatur erstellt wird, aufnehmen; damit wird beweisbar, dass die so eingeschlossene Signatur zum Signaturzeitpunkt bereits existiert hat. Welche Bedeutung allerdings einer Signatur über eine andere Signatur zukommt, ist jedoch, wie bei handschriftlichen Unterschriften auch, vom jeweiligen Kontext abhängig.

Als Mehrfachsignaturen werden im Folgenden Signaturen bezeichnet, die über die selben Daten, also z.B. über das selbe Dokument, erstellt werden. In Abschnitt 2 werden verschiedene Formen von Mehrfachsignaturen – mit und ohne Einschließen bereits existierender Signaturen – betrachtet. Abschnitt 3 zeigt wie Signaturen, die andere Signaturen einschließen, verkettet werden können und gibt unter Punkt 3.1.1 eine Empfehlung zum Verketteten von Signaturen nach dem Konzept Bürgerkarte. In Abschnitt 4 wird diskutiert, wie die Referenzen mehrerer Signaturen so erstellt werden können, dass sie auch tatsächlich die selben Daten signieren, und Abschnitt 5 gibt schließlich Hinweise, was beim Einbetten mehrerer Signaturen in ein Dokument zu beachten ist.

2 Mehrfachsignaturen

Werden über die selben Daten mehrere elektronische Signaturen erstellt, so kann dies grundsätzlich auf zwei verschiedene Arten erfolgen. Elektronische Signaturen über die selben Daten können so erzeugt werden, dass sie voneinander unabhängig sind, oder aber so, dass sie zuvor erzeugte Signaturen einschließen.

Elektronische Signaturen die voneinander unabhängig sind können zeitgleich erzeugt werden und werden daher oft auch als *parallele Signaturen* bezeichnet.

Anwendungsfälle für unabhängige Signaturen über die selben Daten sind zum Beispiel:

- Ein Vertrag wird von den beteiligten Parteien unterzeichnet.
- Ein Dokument wird von mehreren zeichnungsberechtigten Personen unterzeichnet.

Bei voneinander unabhängigen Signaturen gehen in die Signaturerstellung jeweils nur die zu signierenden Daten selbst ein. Normalerweise handelt es sich also um eine Signatur über die selben Daten unter Verwendung unterschiedlicher privater Schlüssel.

Signaturen die andere Signaturen mit einschließen finden z.B. Anwendung bei:

- Gegenzeichnungen

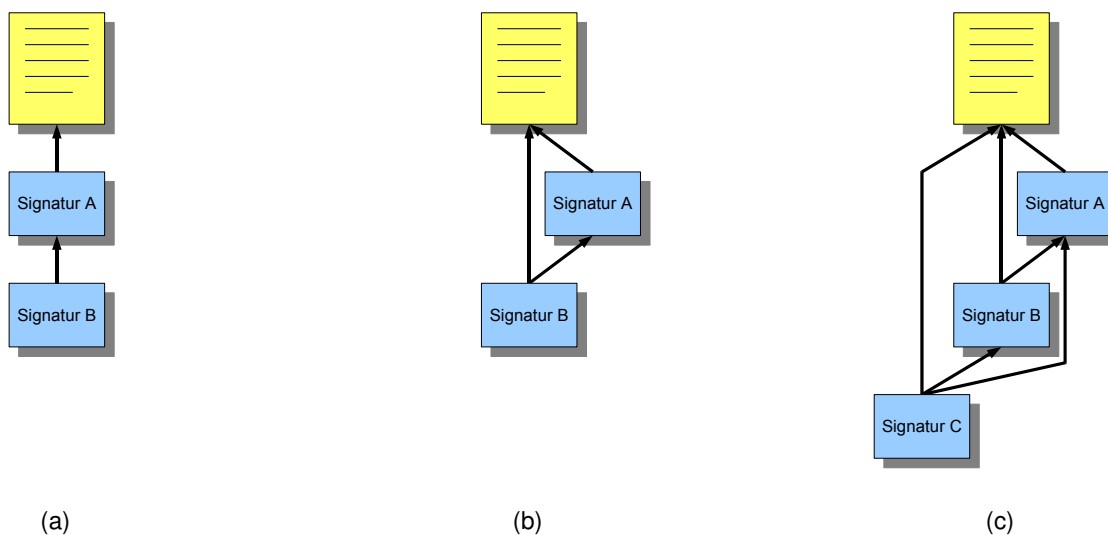
- Beglaubigungen
- Nachsignieren

Welche Bedeutung oder gar rechtliche Wirkung die Signatur einer anderen Signatur entfaltet, hängt, wie bei der handschriftlichen Unterschrift, primär vom Kontext der jeweiligen Anwendung ab. In diesem Dokument werden daher nur die technischen Aspekte der Mehrfachsignatur betrachtet.

3 Verkettungen von Signaturen

Sollen mehrere Signaturen so erstellt werden, dass sie sowohl die zu signierenden Daten als auch andere bereits existierende Signaturen einschließen, dann kann dies aus technischer Sicht auf verschiedene Arten erfolgen.

Grundsätzlich besteht, wie in Abb. 1(a) gezeigt, bei einer Signatur B , die eine Signatur A signiert eine kryptographische Bindung zwischen der Signatur B und den durch die Signatur A signierten Daten.¹ Ist die Signatur A jedoch ungültig, besteht allerdings auch keine kryptographische Bindung zwischen der Signatur B und den durch die Signatur A signierten Daten. Warum die Signatur A als ungültig betrachtet wird, spielt dabei keine Rolle.



$B \rightarrow A \quad \dots \quad B \text{ signiert } A$

Abb. 1: Verkettungen von elektronischen Signaturen

¹ Jede Änderung der Daten hätte die Ungültigkeit der Signatur A zur Folge. Würde die Signatur A entsprechend über die geänderten Daten erstellt, dann würde dadurch die Signatur B ungültig werden.

Um die kryptographische Bindung, zwischen den signierten Daten und der Signatur *B*, unabhängig von der Gültigkeit der Signatur *A*, zu gewährleisten, sollte, wie in Abb. 1(b) gezeigt, die Signatur *B* auch direkt über die von Signatur *A* signierten Daten erstellt werden. Sind, wie in Abb. 1(c) gezeigt, bereits mehrere Signaturen vorhanden, dann müssen, zur Vermeidung von Abhängigkeiten, alle weiteren Signaturen (Signatur *C*) jeweils über die zu signierenden Daten und *alle* bereits vorhandenen Signaturen (Signatur *A* und Signatur *B*) erstellt werden.

3.1 Verketteten von Signaturen nach dem Konzept Bürgerkarte

In der folgenden Diskussion werden Signaturen nach dem *XML Digital Signatures* Standard [1] betrachtet, da sie flexible Möglichkeiten zum Verketteten und Referenzieren von Teilen eines Dokuments bieten und im österreichischen E-Government am häufigsten Verwendung finden.

Eine der Stärken von *XML Digital Signatures* (XMLDSig) ist ein sehr mächtiger Mechanismus zur Spezifikation der Daten, die in die Signaturerstellung einfließen. Damit können gezielt die zu signierenden Teile eines Dokuments referenziert und beliebige Transformationen darauf angewendet werden. Unterstützt werden dabei normalerweise:

- URI Referenzen nach XPointer [2]
- XPath Filtering [1]
- XPath Filter 2.0 [3]
- XSLT Transformationen [1]
- Enveloped Signature Transformationen [1]
- Canonical XML [4] und Exclusive XML Canonicalization Transformationen [5]

Dieser Mechanismus ermöglicht es die zu signierenden Daten und bereits existierende Signaturen flexibel zu referenzieren und darüber die Signatur zu berechnen.

Um eine kryptographische Bindung zwischen der einschließenden Signatur und der eingeschlossenen Signatur zu erreichen, ist es zumindest erforderlich, dass die einschließende Signatur über den Signaturwert (also den Inhalt des `SignatureValue`-Elements) der eingeschlossenen Signatur erstellt wird. Es ist also nicht erforderlich, die ganze Datenstruktur einer bestehenden Signatur in die einschließende Signatur aufzunehmen, um eine kryptographische Bindung herzustellen.

3.1.1 Verketteten mittels Signaturblock

Zum Verketteten von Signaturen nach dem Konzept Bürgerkarte, wird ein Mechanismus empfohlen, der eine Anzeige der eingeschlossenen Signatur als Signaturblock ähnlich dem Signaturblock der Amtssignatur erlaubt. Dazu wird ein `Reference`-Element so erstellt, dass es das `Signature`-Element und alle seine Nachfahren referenziert und durch eine XSL-Transformation eine XHTML-Repräsentation der Signatur erzeugt, die zumindest folgende Elemente enthält:

- Seriennummer des Zertifikats, sowie Name und Herkunftsland des Zertifizierungsdiensteanbieters, der das Zertifikat ausgestellt hat²
- Signaturwert (Inhalt des `SignatureValue`-Elements)

Zusätzlich können auch noch weitere Elemente, wie z.B. der Zeitpunkt der Signatur oder der Name des Antragstellers des Zertifikats³ dargestellt werden.

Signaturzeitpunkt		2006-11-16T16:18:50Z
Signaturwert		d3vIkJNfXzevje227AGrcBaT/XZNgtDtF5lmj0MxNwuVPzQoETYPal8uRj5Pg3/6
Zertifikat	Aussteller	C=AT,O=Hauptverband österr. Sozialvers.,CN=VSig CA 2
	Seriennummer	17229045246817736659347185373920056355859

Abb. 2: Beispiel Signaturblock

Zur Kennzeichnung der Referenz auf eine bestehende Signatur sollte die Spezifikation der Bürgerkarte eine URI definieren, die als Wert des `Type`-Attributs der `Reference` gesetzt werden muss. Damit ist es Applikationen leicht möglich die entsprechenden Referenzen zu identifizieren.

Enthält die `Reference` ein XSL-Stylesheet zur Transformation der referenzierten Signatur, dann wird die Signatur entsprechend [1] 4.3.3.2 (*The Reference Processing Model*) über die transformierten Daten, also über den Signaturblock, erstellt. Damit eine Applikation die Möglichkeit erhält automatisch zu prüfen, ob tatsächlich die relevanten Daten der referenzierten Signatur signiert wurden, wird es in der Praxis erforderlich sein, ein Stylesheet zur Transformation zu verwenden, bei dem die Applikation auf ein korrektes Transformationsergebnis vertrauen kann.

Die Signatur-Applikation sollte, beim Verketteten von Signaturen, neben der erforderlichen Anzeige des Signaturblocks, zusätzlich die Möglichkeit bieten, die referenzierte Signatur zu prüfen und die damit signierten Daten darzustellen.⁴ Die Signatur *B* über eine Signatur *A* sagt allerdings normalerweise nichts über die Gültigkeit der Signatur *A* aus. Eine Prüfung der Signatur *A* bei der Erstellung der Signatur *B* wird also technisch keineswegs vorausgesetzt oder gefordert – in der Anwendung wird dies aber oft notwendig sein (z.B. bei Beglaubigungen).

Beim Verketteten von Detached-Signatures ([1] 10.0 *Definitions*) die in einem getrennten Dokument gespeichert werden, kann eine XSL-Transformation zum Erzeugen des Signaturblocks ebenfalls eingesetzt werden.⁵

² `X509IssuerName` und `X509SerialNumber` sind geeignet um das Zertifikat des Unterzeichners eindeutig zu identifizieren und können aus dem entsprechenden `SigningCertificate`-Element übernommen werden.

³ Dies ist jedoch nicht über eine einfache XSL-Transformation möglich, da die entsprechenden Daten nur im Zertifikat und nicht in der Signatur enthalten sind.

⁴ Zu beachten ist dabei, dass zur Prüfung direkt die referenzierte Signatur und nicht die Input-Daten für die Stylesheet Transformation zur Erzeugung des Signaturblocks verwendet werden, da die nach [1] 4.3.3.2 (*The Reference Processing Model*) notwendige Kanonisierung [4] möglicherweise die Signatur bricht.

⁵ Zu beachten ist dabei jedoch, dass eine Referenz auf ein anderes Dokument (nicht-*reference-only* URI-Referenz) nach [1] in Octets aufgelöst wird. Ist das `Signature`-Element nicht das Wurzelement des referenzierten Doku-

3.1.2 Alternative Methoden zum Verketteten von Signaturen

Ist keine Anzeige der eingeschlossenen Signatur erforderlich, können auch alternative Methoden zum Verketteten gewählt werden. Jedenfalls muss zum Verketteten der Signaturwert, also der Wert des `SignatureValue`-Elements eingeschlossen werden. Handelt es sich bei der eingeschlossenen Signatur A um eine Signatur nach dem Konzept Bürgerkarte, ist darauf zu achten, dass diese Elemente der XML Advanced Electronic Signature (XAdES) Spezifikation [6] enthält. Damit sollte die Signatur A durch die Signatur B jedenfalls so referenziert werden, dass entsprechend der XAdES-Spezifikation, später weitere Elemente (`UnsignedProperties`) zu den `QualifyingProperties` der Signatur A hinzugefügt werden können, ohne dass die Signatur B bricht.

4 Referenzieren der Daten

Wie unter Punkt 3 diskutiert, besitzt XMLDSig einen sehr mächtigen Mechanismus zum Referenzieren der zu signierenden Daten. Sollen in einer Applikation Mehrfachsignaturen zum Einsatz kommen können, dann müssen Referenzen mehrerer Signaturen so erzeugt werden können, dass sie jeweils die selben Daten referenzieren. Sind die zu signierenden Daten aus dem Applikationskontext bekannt, so können die entsprechenden Referenzen einfach erzeugt werden. Sind jedoch aus dem Applikationskontext die zu signierenden Daten nicht bekannt, dann muss aus den Referenzen der existierenden Signaturen auf die signierten Daten, und damit auf die Referenzen der zu erstellenden Signatur, geschlossen werden.

Die Referenzen auf die zu signierenden Daten werden in XMLDSig innerhalb des `SignedInfo`-Blocks zusammengefasst. Um eine Signatur B zu erstellen, die exakt die selben Daten signiert wie eine Signatur A , ist es nötig einen `SignedInfo`-Block zu erzeugen, der zu jeder `Reference` aus der Signatur A eine `Reference` für die Signatur B enthält, die unter Anwendung des gleichen Hash-Algorithmus den gleichen Hash-Wert ergibt, d.h. deren Ergebnisse nach der Verarbeitung nach [1] 4.3.3.2 (*The Reference Processing Model*) identisch sind. Diese Anforderung ist teilweise sehr schwer zu erfüllen, insbesondere, wenn in der `Reference` relative URIs, `XPointer` oder `XPath`-Ausdrücke bzw. kontextabhängige Funktionen wie z.B. die `XPath here()`-Funktion enthalten sind. Es wird daher im Allgemeinen nicht möglich sein, einen Algorithmus zu formulieren, der zu jeder Referenz aus einer Signatur A eine Referenz einer Signatur B erzeugt, die bei entsprechender Verarbeitung das selbe Ergebnis liefert.

Um den Einsatz von Mehrfachsignaturen zu erleichtern, sollte daher schon bei der Applikationsentwicklung darauf geachtet werden, dass die Referenzen der entsprechenden Signaturen einfach gebildet werden können.

Alternativ zum Erstellen entsprechender Referenzen auf die selben Daten in jeder Signatur kann auch ein Manifest ([1] 5.1) eingesetzt werden. Die zu signierenden Daten werden nur einmal durch

ments, muss das `Signature`-Element über eine entsprechende Transformation selektiert werden, bevor die XSLT-Transformation angewendet werden kann. Ist das `Signature`-Element jedoch Wurzelement des referenzierten Dokuments, kann die XSLT-Transformation direkt angewandt werden, da sie als Input-Daten ohnehin Octets fordert.

das Manifest referenziert. Das Manifest wird wiederum durch entsprechende Referenzen in allen Signaturen referenziert (Abb. 3(a)). Jede Signatur kann auch noch weitere Referenzen enthalten, z.B. auf verkettete Signaturen (Abb. 3(b)), oder im Falle von Signaturen nach dem Konzept Bürgerkarte auf `SignedProperties` [6]. Die Referenzen auf das Manifest sollten dabei zur Kennzeichnung als Type-Attribut die URI `http://www.w3.org/2000/09/xmldsig#Manifest` enthalten.⁶ Bei der Signaturprüfung müssen dann auch zusätzlich die Referenzen des Manifests geprüft werden.



(a)

(b)

$$B \longrightarrow A \quad \dots \quad B \text{ signiert } A$$

Abb. 3: Signatur über die selben Daten mit Manifest

Diese Variante eignet sich allerdings nur bedingt zum Nachsignieren, da für die signierten Daten bei erneuter Signatur kein neuer Hash-Wert berechnet wird. Damit schützt das Nachsignieren in diesem Fall nicht vor einem zu schwachen Hash-Algorithmus.

Zu beachten ist weiters, dass vor dem Erstellen weiterer Signaturen die Hash-Werte im Manifest geprüft werden sollten, um sicher zu stellen, dass sich die erstellten Signaturen auch tatsächlich auf die referenzierten Daten beziehen.

5 Einbetten mehrerer Signaturen in ein Dokument

Generell ist beim Einbetten von Signaturen in ein Dokument zu beachten, dass die Signatur im Kontext vom Dokument nicht in jedem Fall verifizierbar bleibt, wenn die Signatur ohne diesen Kontext erstellt wurde. Vom Kontext, in den die Signatur eingebettet ist werden Informationen geerbt,

⁶ Das `Manifest`-Element ist dabei (zumindest für alle weiteren Signaturen) außerhalb des `Signature`-Elementes. Manche Signatur-Bibliotheken bzw. Applikationen unterstützen externe Manifeste – obwohl in XMLDSig [1] vorgesehen – jedoch nicht.

die dann in die Hash- oder Signatur-Berechnung mit einfließen können. Die Signatur ist dabei in ihrem ursprünglichen Kontext (z.B. extern vom Dokument) weiter verifizierbar, in dem Kontext in dem sie eingebettet wird aber möglicherweise nicht.

Wird eine eingebettete Signatur A nun in eine Signatur B eingeschlossen, dann wird möglicherweise auch der Kontext der Signatur A mit eingeschlossen. Wurde die Signatur A ohne diesen Kontext erstellt, so ist die Signatur B über die Signatur A nur verifizierbar, wenn die Signatur A ins Dokument eingebettet ist, die Signatur A selbst ist jedoch nur verifizierbar, ohne diesen Kontext.

Daher sollten Signaturen generell so erstellt werden, dass sie unabhängig von dem Kontext sind, in den sie eingebettet werden. Beim Verketteten von eingebetteten Signaturen sollte zusätzlich darauf geachtet werden, dass keine Informationen vom Kontext der eingebetteten Signatur in die Signaturerstellung mit einfließen. Beim Verketteten von Signaturen über einen mit einer XSL-Transformation erstellten Signaturblock (3.1.1) sollte daher vor der XSLT-Transformation eine Exclusive XML Canonicalization Transformation [5] eingefügt werden, um keine Namespace-Informationen aus dem Kontext der eingebetteten Signatur in das Transformationsergebnis mit aufzunehmen.⁷

6 Zusammenfassung

In Abschnitt 3.1.1 wird ein Mechanismus zum Verketteten von Signaturen nach dem Konzept Bürgekarte empfohlen, der die Anzeige der zu signierenden Signatur in Form eines Signaturblocks, ähnlich dem der Amtssignatur, erlaubt. Dadurch wird einerseits ein kryptographisches Verketteten mit der zur signierenden Signatur und andererseits die Anforderung der Darstellbarkeit der signierten Daten erreicht.

Abhängig von der Art und Weise, wie die zu signierenden Daten referenziert und die erzeugten Signaturen in ein Dokument eingebettet werden sollen, sind beim Entwickeln von Anwendungen, die Mehrfachsignaturen unterstützen sollen, weiters auch die in Abschnitt 4 und 5 diskutierten Aspekte zu berücksichtigen, um die Integration von Mehrfachsignaturen zu ermöglichen und eine entsprechende Robustheit zu erhalten.

⁷ Da der Input für die XSLT-Transformation nach [1] als Octets definiert ist, findet bei einem Node-Set als Input-Daten automatisch eine Kanonisierung nach Canonical XML [4] statt, bei der die Namespace-Informationen aus dem Kontext unabhängig von der tatsächlichen Verwendung des Namespace-Präfix geerbt werden. Die zusätzlich eingefügte Exclusive XML Canonicalization Transformation bedeutet daher auch keinen Mehraufwand bei der Verarbeitung der Referenz.

A Beispiel für ein Reference-Element zur Verkettung von Signaturen

Das folgende Listing zeigt beispielhaft ein Reference-Element zum Verkettung von Signaturen mit entsprechendem XSL-Stylesheet zur Erzeugung eines Signaturblocks entsprechend Punkt 3.1.1.

Das URI-Attribut in Zeile 4 referenziert auf das Signature-Element der eingeschlossenen Signatur. Der Wert für das Type-Attribut in Zeile 3 muss noch spezifiziert werden und ist hier nur als Beispiel gegeben.

Die Transformation in Zeile 6 wurde entsprechend den Überlegungen des Abschnitts 5 eingefügt. Das in der Transformation in den Zeilen 7 bis 53 enthaltene XSL-Stylesheet erzeugt den Signaturblock aus Abbildung 2.

```

1  ...
2  <dsig:Reference Id="signature-reference-0-1163693930-78328069-17717"
3    Type="http://www.buergerkarte.at/specifications/securitylayer/..."
4    URI="#signature-1163693930-78328069-29799">
5    <dsig:Transforms>
6      <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
7      <dsig:Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
8        <xsl:stylesheet version="1.0" xml:space="preserve"
9          xmlns="http://www.w3.org/1999/xhtml"
10         xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
11         xmlns:etsi="http://uri.etsi.org/01903/v1.1.1#"
12         xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
13          <xsl:output method="xml" />
14          <xsl:template match="/">
15            <html>
16              <head>
17                <title>Signaturdaten</title>
18              </head>
19              <body>
20                <table border="1">
21                  <tbody>
22                    <tr>
23                      <td colspan="2" align="center">Signaturzeitpunkt</td>
24                      <td style="break-all; word-wrap: break-word">
25                        <xsl:value-of select="//dsig:Object/etsi:QualifyingProperties/etsi:
26                          SignedProperties/etsi:SignedSignatureProperties/etsi:SigningTime"/>
27                      </td>
28                    </tr>
29                    <tr>
30                      <td colspan="2" align="center">Signaturwert</td>
31                      <td style="word-break: break-all; word-wrap: break-word">
32                        <xsl:value-of select="translate(normalize-space(//dsig:SignatureValue),' ','')
33                          "/>
34                      </td>
35                    </tr>
36                    <tr>
37                      <td rowspan="2" align="center">Zertifikat</td>
38                      <td align="center">Aussteller</td>
39                      <td>
40                        <xsl:value-of select="//dsig:Object/etsi:QualifyingProperties/etsi:
41                          SignedProperties/etsi:SignedSignatureProperties/etsi:SigningCertificate/
42                          etsi:Cert/etsi:IssuerSerial/dsig:X509IssuerName"/>
43                      </td>
44                    </tr>
45                    <tr>
46                      <td align="center">Seriennummer</td>
47                      <td style="word-break: break-all; word-wrap: break-word">
48                        <xsl:value-of select="//dsig:Object/etsi:QualifyingProperties/etsi:
49                          SignedProperties/etsi:SignedSignatureProperties/etsi:SigningCertificate/
50                          etsi:Cert/etsi:IssuerSerial/dsig:X509SerialNumber"/>
51                      </td>
52                    </tr>
53                  </tbody>
54                </table>
55              </body>
56            </html>
57          </template>
58        </dsig:Transform>
59      </dsig:Transforms>
60    </dsig:Reference>

```

```
45         </td>
46     </tr>
47 </tbody>
48 </table>
49 </body>
50 </html>
51 </xsl:template>
52 </xsl:stylesheet>
53 </dsig:Transform>
54 <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
55 </dsig:Transforms>
56 <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
57 <dsig:DigestValue>JSc4YNBlY6QZD70a6XgHiiRysLs=</dsig:DigestValue>
58 </dsig:Reference>
59 ...
```

Änderungen

Version 1.0 24.11.2006 Martin Centner

Version 1.0.1 29.11.2006 Martin Centner

Editorielle Änderungen

Referenzen

- [1] EASTLAKE, Donald E. ; REAGLE, Joseph M. ; SOLO, David: *XML-Signature Syntax and Processing*. World Wide Web Consortium, Recommendation REC-xmldsig-core-20020212, February 2002
- [2] GROSSO, Paul ; MALER, Eve ; MARSH, Jonathan ; WALSH, Norman: *XPointer Framework*. World Wide Web Consortium, Recommendation REC-xptr-framework-20030325, March 2003
- [3] BOYER, John M. ; HUGHES, Merlin ; REAGLE, Joseph M.: *XML-Signature XPath Filter 2.0*. World Wide Web Consortium, Recommendation REC-xmldsig-filter2-20021108, November 2002
- [4] BOYER, John M.: *Canonical XML Version 1.0*. World Wide Web Consortium, Recommendation REC-xml-c14n-20010315, March 2001
- [5] BOYER, John M. ; EASTLAKE, Donald E. ; REAGLE, Joseph M.: *Exclusive XML Canonicalization Version 1.0*. World Wide Web Consortium, Recommendation REC-xml-exc-c14n-20020718, July 2002
- [6] CRUELLAS, Juan C. ; KARLINGER, Gregor ; PINKAS, Denis ; ROSS, John: *XML Advanced Electronic Signatures (XAdES)*. World Wide Web Consortium, Note NOTE-XAdES-20030220, February 2003