

Kryptologie, Prof. Köhler

# XML-Signatur

Frank Markus Abbühl

20. Juli 2004

Fachhochschule München

## Inhaltsverzeichnis

<b>1</b>	<b>Rationale</b>	<b>3</b>
<b>2</b>	<b>Standard</b>	<b>3</b>
2.1	Anforderungen . . . . .	3
2.2	Syntax . . . . .	4
<b>3</b>	<b>Verarbeitungsvorschriften</b>	<b>5</b>
3.1	Kanonisierung . . . . .	6
3.2	Erzeugung . . . . .	6
3.3	Validierung . . . . .	7
<b>4</b>	<b>Anwendungs-Entwicklung</b>	<b>7</b>
4.1	Sicherheit . . . . .	7
4.2	Implementierungen . . . . .	7

## **Abkürzungsverzeichnis**

- ASN.1** Abstract Syntax Notation One
- C14N** Canonicalization
- CMS** Cryptographic Message Syntax Standard
- DOM** Document Object Model
- DTD** Document Type Definition
- IPSec** IP Security
- P3P** Platform for Privacy Preferences
- PGP** Pretty Good Privacy
- PKCS** Public Key Cryptography Standard
- PKI** Public Key Infrastructure
- RFC** Request For Comment
- SSL** Secure Socket Layer
- SAML** Security Assertion Markup Language
- S/MIME** Secure Multipurpose Internet Mail Extensions
- TLS** Transport Layer Security
- UML** Unified Modeling Language
- URI** Uniform Resource Identifier
- W3C** World Wide Web Consortium
- WSS** Web Services Security
- XACML** XML Access Control Markup Language
- XKMS** XML Key Management Specification
- XML** eXtensible Markup Language
- XMLDSIG** XML Digital Signature
- XMLENC** XML Encryption
- XPath** XML Path Language
- XrML** eXtensible Rights Markup Language

# 1 Rationale

Mit den Secure Multipurpose Internet Mail Extensions (S/MIME) und Pretty Good Privacy (PGP) existieren mindestens zwei etablierte Mechanismen für die Verschlüsselung und Digitale Signatur von Dokumenten. Zur Absicherung von Datenkommunikation auf verschiedenen Protokollebenen können Protokollzusätze wie IP Security (IPSec) oder Secure Socket Layer (SSL) beziehungsweise Transport Layer Security (TLS) eingesetzt werden. Warum sollte man sich mit einem weiteren kryptografischen Standard beschäftigen, nur weil dieser auf der in Mode gekommenen eXtensible Markup Language (XML) basiert?

Die XML selbst gewinnt zunehmend an Akzeptanz in der Industrie. Dem Programmierer stehen ausgereifte Parser und frei verfügbare Software-Bibliotheken zur Verfügung. Moderne, textorientierte Kommunikationsprotokolle und Dokumentformate lösen nach und nach ihre binären Vorgänger ab. Bestehende Sicherheitssoftware ist schwer in reine XML-Umgebungen zu integrieren: Binärformate für Nachrichten und Zertifikate<sup>1</sup>, fehlende Unterstützung für Internet-Verweise und die Abhängigkeit von herstellerspezifischen Software-Bibliotheken erschweren die Arbeit des Programmierers.

## 2 Standard

Seit Anfang 1999 beschäftigt sich eine Arbeitsgruppe des World Wide Web Consortium (W3C) mit der Entwicklung von XML-konformen Standards für Signatur und Verschlüsselung. Im Laufe des Jahres 2002 veröffentlichte das W3C mehrere Empfehlungen zu diesem Thema. Die hier vorgestellte Empfehlung [XML-Signature] beschreibt Syntax und Verarbeitungsvorschriften bei der Erstellung und Validierung von Digitalen Signaturen.

### 2.1 Anforderungen

Die Empfehlung schreibt keine Interpretation der Signatur vor, etwa die Zuordnung von Schlüsseln zu Personen. Sie legt sich nicht auf eine Public Key Infrastructure (PKI) fest, sie definiert keine neues Schema für Zertifikate und sie macht auch keine Aussagen über Vertrauensbeziehungen. Die XML-Signatur stellt ausschließlich einen Bezug zwischen einem Schlüssel und beliebigen referenzierten Daten her.

Man wollte einen guten Kompromiß zwischen Einfachheit und Flexibilität finden, um dem Entwickler die Umsetzung beliebig komplexer Szenarien zu ermöglichen. Die XML-Signatur stellt die Authentizität und Integrität von beliebigen Datenobjekten sicher, egal ob es sich um Binärdaten, Textdaten, XML-Dokumente oder -Dokumentteile handelt. Erlaubt ist alles, worauf man per Uniform Resource Identifier (URI) verweisen kann. Bei der Validierung verhält sie sich robust gegenüber den Variabilitäten der Darstellung von XML. Die Verwendung beliebiger kryptografischer Algorithmen ist dem Entwickler freigestellt.

---

<sup>1</sup>S/MIME basiert auf PKCS#7 und X.509-Zertifikaten, man benötigt einen ASN.1-Parser zur Dekodierung

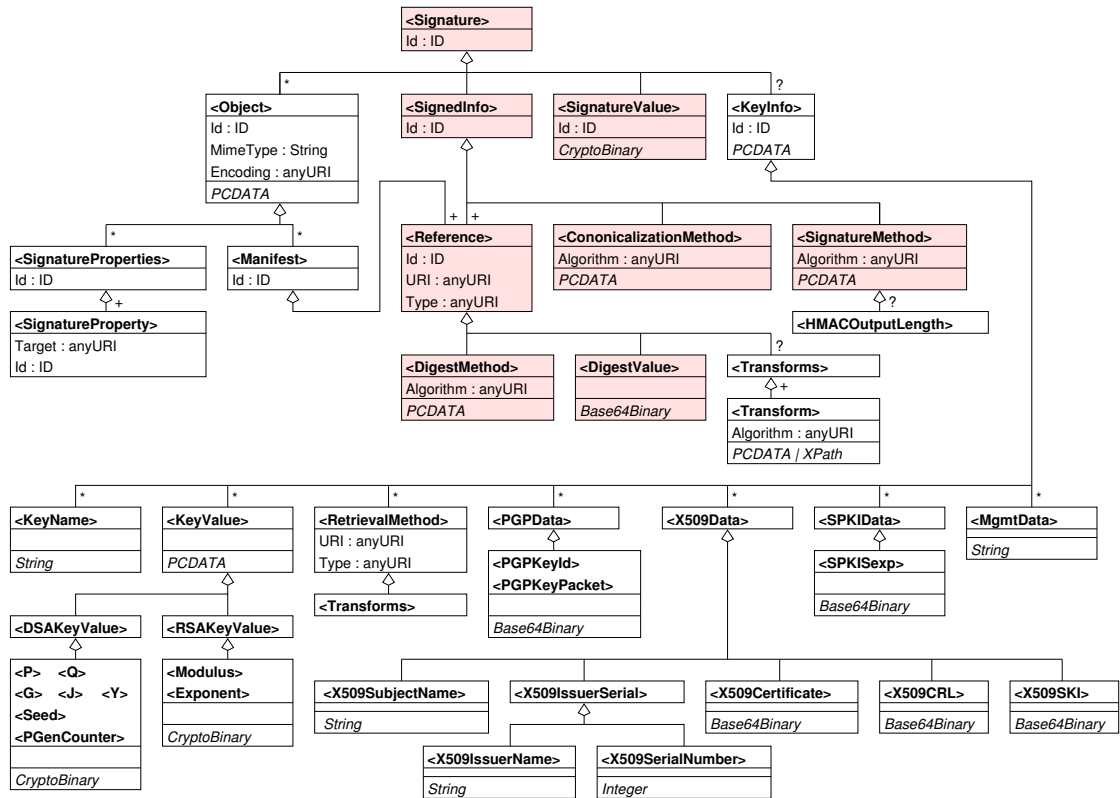


Abbildung 1: Schema-Definition der XML-Signatur

## 2.2 Syntax

Die XML-Signatur ist ein XML-Element mit einer bestimmten Syntax. Sie besteht aus einem Manifest, welches Verweise auf die signierten Datenobjekte sowie deren Hash-Werte enthält, und dem Signatur-Wert. Optional kann sie Schlüsselinformationen und beliebig viele weitere Datenobjekte beinhalten.

Die Abbildung 1 zeigt eine vollständige Darstellung der Schema-Definition der XML-Signatur in einer an die Unified Modeling Language (UML) angelehnten Notation. Anstatt einer umfassenden, formalen Beschreibung der einzelnen Elemente muß hier ein einfaches Beispiel genügen, einen Eindruck für die äußere Form einer XML-Signatur zu vermitteln. Die Details lassen sich leicht in [XML-Signature] nachlesen.

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
    <ds:SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1" />
    <ds:Reference URI="payment.xml">
```

```

    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
    <ds:DigestValue>E71h260mCduj9rgFnUm0cw4Uu7Y=</ds:DigestValue>
  </ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>
  U3+JLn7CsHvFjx+sbbft7ESDxCBtsZ7V6F7mveA8/7idl9wioAyKig==
</ds:SignatureValue>
</ds:Signature>

```

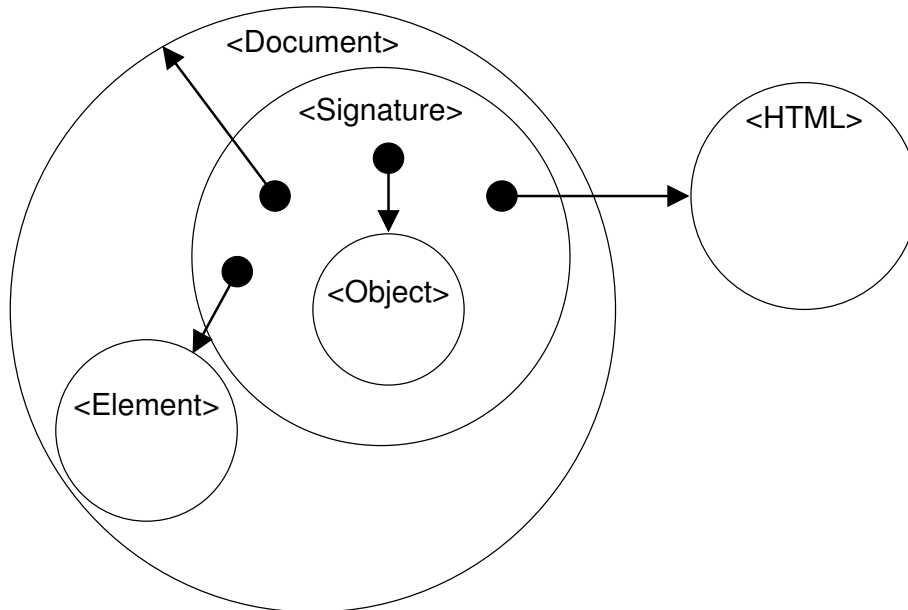


Abbildung 2: Alleinstehende, umhüllte und umhüllende Signatur

Im einfachsten Fall verweist eine Referenz auf ein externes Dokument oder auf ein von der XML-Signatur disjunktes XML-Element. Man nennt dies eine *alleinstehende Signatur* (detached signature). Von einer *umhüllten Signatur* (enveloped signature) spricht man, wenn eine Referenz auf ein XML-Element verweist, welches die Signatur enthält. Die dritte Möglichkeit besteht darin, dass eine Referenz auf ein Objekt-Element innerhalb der Signatur verweist. Diese Variante heißt *umhüllende Signatur* (enveloping signature). Diese Varianten können, wie in Abbildung 2 dargestellt, beliebig kombiniert werden.

### 3 Verarbeitungsvorschriften

Neben der Syntax legt die Empfehlung auch Verarbeitungsvorschriften für die Erzeugung und Validierung der Digitalen Signatur fest. Kanonisierung sorgt dabei für die erforderliche Robustheit.

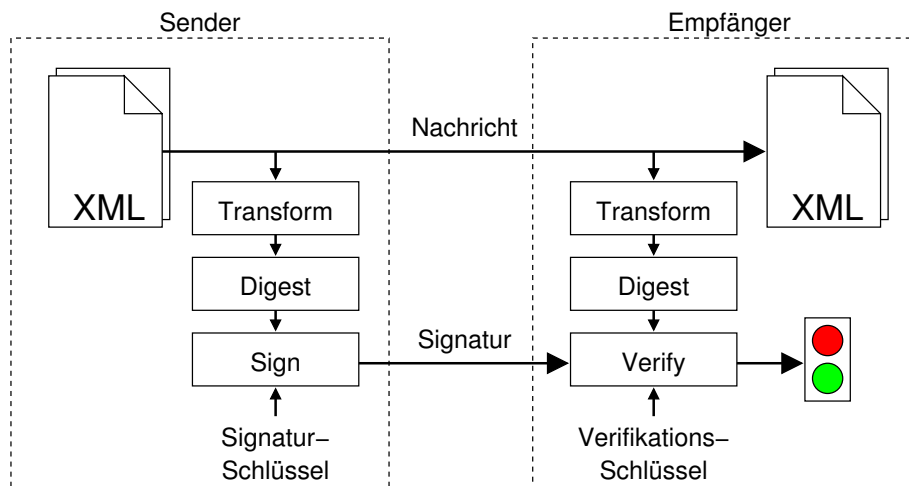


Abbildung 3: Signatur-Protokoll

### 3.1 Kanonisierung

Die XML kennt keine Standardform für die Darstellung von Dokumenten<sup>2</sup>. Darüberhinaus konkurrieren verschiedene Datenmodelle miteinander, zum Beispiel das Document Object Model (DOM) und die XML Path Language (XPath). Und schlimmer noch, die XML-Repräsentation selbst ist veränderlich. Beispielsweise ist die Reihenfolge von Attributen innerhalb eines Elements irrelevant.

Es wird also ein Mechanismus benötigt, der XML-Daten auf eindeutige Weise serialisiert, so dass diese durch kryptografische Algorithmen verarbeitet werden können. Beide Probleme werden durch Kanonisierung gelöst. Die Empfehlung XML-C14N beschreibt, wie insignifikante Aspekte entfernt und die Standardform hergestellt wird. Die XML-Deklaration sowie eine eventuell eingebettete Document Type Definition (DTD) werden entfernt. Entitäts-Referenzen werden durch ihre Definition ersetzt. Zeilenumbrüche werden vereinheitlicht. CDATA Abschnitte werden aufgelöst und Reihenfolge und Inhalt von Attributen werden normalisiert, Standardattribute werden expandiert. Zum Schluß wird das Dokument in eine UTF-8 kodierte Zeichenkette umgewandelt.

### 3.2 Erzeugung

Eine XML-Signatur entsteht in zwei Schritten. Im Ersten werden die Referenz-Elemente erzeugt. Dazu werden alle angegebenen Transformationen auf die referenzierten Datenobjekte angewandt, einschließlich der Kanonisierung im Falle von XML-Daten. Über die transformierten Daten wird jeweils der Hash-Wert mit Hilfe des im `<DigestMethod>` Element angegebenen Algorithmus berechnet und im `<DigestValue>` Element abgespeichert.

<sup>2</sup>Die Speicherung als Folge von Zeichen in einer Datei auf der Festplatte ist nur eine Möglichkeit, eine andere ist die Repräsentation als DOM-Baum im Arbeitsspeicher

Im Zweiten Schritt wird der Signatur-Wert berechnet. Zuerst wird das `<SignedInfo>` Element kanonisiert, unter Verwendung des im `<CanonicalizationMethod>` Element angegebenen Algorithmus. Über die resultierenden Daten wird der Hash-Wert gebildet. Mit Hilfe des im `<SignatureMethod>` Element angegebenen Algorithmus wird dann der Signatur-Wert berechnet und im `<SignatureValue>` Element gespeichert.

### 3.3 Validierung

Analog zur Signatur-Erstellung verläuft auch die Validierung in zwei Schritten: Zuerst wird die Integrität der referenzierten Datenobjekte sichergestellt. Das `<SignedInfo>` Element wird kanonisiert. Die zu jedem `<Reference>` Element gehörenden Datenobjekte werden transformiert und gehasht. Die berechneten Hash-Werte werden mit dem Inhalt des entsprechenden `<DigestValue>` Elements verglichen. Die Validierung schlägt fehl, sobald Unstimmigkeiten auftreten.

Anschließend überprüft man die Authentizität der Signatur. Das kanonisierte `<SignedInfo>` Element wird erneut gehasht und der Signatur-Wert wird berechnet. Stimmt der berechnete Wert mit dem Inhalt des `<SignatureValue>` Elements überein, ist die Signatur authentisch.

## 4 Anwendungs-Entwicklung

### 4.1 Sicherheit

Wie anfangs bereits erwähnt, stellt die XML-Signatur lediglich einen Bezug her zwischen einem Schlüssel und den referenzierten Daten. Aufgabe des Entwicklers ist es, diesen Bezug mit Bedeutung zu erfüllen. Ebenfalls obliegt ihm die Auswahl geeigneter kryptografischer Algorithmen sowie die syntaktische Einpassung der XML-Signatur in seine Dokumente oder Protokolle.

Die hohe Flexibilität der XML-Signatur birgt auch Risiken. Durch das Anwenden von XPath-Filtern oder anderen Transformationen können gezielt Passagen eines referenzierten Dokuments ausgeblendet werden. Eine Manipulation dieser ausgeblendeten Passagen invalidiert demnach nicht die Signatur. *Nur was signiert ist, ist auch sicher.* Das Anwendungsprogramm soll dem Anwender daher in geeigneter Weise die tatsächlich durch die Signatur abgedeckten Daten präsentieren.

Ebenfalls gefährlich ist es, Daten unbesehen zu signieren, denn damit verliert die Signatur ihre eigentliche Bedeutung. Es gilt der Grundsatz: *Nur was man sieht, soll man unterzeichnen.* Das Anwendungsprogramm soll dem Anwender die tatsächlich zu unterzeichnenden Daten in geeigneter Weise präsentieren und die Unterschrift so zu einem bewußten Akt machen.

### 4.2 Implementierungen

Kein namhafter Anbieter von Sicherheits-Software hat es versäumt, die Welt um eine eigene Implementierung dieses Standards zu bereichern. Entsprechend groß ist die Auswahl

für den Anwendungs-Entwickler. Die Angaben aus Tabelle 4.2 entstammen größtenteils Eastlake and Niles [2002, Appendix A] und wurden durch aktuelle Herstellerinformationen aus dem Internet ergänzt.

Hersteller	Name	Sprachen	Lizenz
Aleksey	XML Security Library	C	MIT (OSS)
Apache	XML Security	Java, C++	Apache (OSS)
Clarios	XML Signature/Enc.	Java, C++	royalty free
Entrust	Security Toolkit	Java	k.A.
IAIK	XML Signature Library	Java	800 €
IBM	XML Security Suite	Java	1000 US\$
Infomosaic	SecureXML	.NET	k.A.
Microsoft	SignedXml	.NET	Teil der Klassenbibliothek
NEC	XMLDSIG	Java	public domain
NeuClear	XMLSig	Java	GPL (OSS)
Phaos	Phaos XML	Java	k.A.
RSA Security	BSAFE Cert-J	Java	k.A.
SETCCE	XSign	COM/ActiveX	1940 €
Ubisecure	Ubisignature	COM/ActiveX	k.A.
Uni Pisa	Gapxse	Java	LGPL (OSS)
Verisign	XML Signature	Java	k.A.

Mittlerweile entsteht ein regelrechter Standard-Dschungel rund um XML-Signatur und -Verschlüsselung. In einem einführenden Artikel rund um das Thema XML-Sicherheit beschreibt Hirsch [2002] die darauf basierende XML Key Management Specification (XKMS) zur Schlüssel-Verwaltung, die Security Assertion Markup Language (SAML) zur Authentifikation von Anwendern und die XML Access Control Markup Language (XACML) zur Autorisierung beim Zugriff auf Ressourcen. Anwendung finden diese Technologien bei der Sicherung von Online-Diensten mittels Web Services Security (WSS), im persönlichen Datenschutz durch die Platform for Privacy Preferences (P3P) oder zur Rechte-Verwaltung mit der eXtensible Rights Markup Language (XrML).

## Literatur

Donald E. Eastlake and Kitty Niles. *Secure XML*. Pearson Professional Education, 2002.

Frederick Hirsch. Getting Started With XML Security, 2002. URL <http://www.fjhirsch.com/xml/xmlsec/starting-xml-security.html>.

XML-C14N. Canonical XML Version 1.0, March 2001. URL <http://www.w3.org/TR/xml-c14n>.

XML-Signature. XML-Signature Syntax and Processing, Februar 2002. URL <http://www.w3.org/TR/xmlsig-core/>.