Signaturformate für elektronische Rechnungen

Detlef Hühnlein[1] und Ulrike Korte[2]

[1] secunet Security Networks AG,
Sudetenstraße 16, 96247 Michelau,
detlef.huehnlein@secunet.com

[2] Bundesamt für Sicherheit in der Informationstechnik,
Godesberger Allee 185 - 189, 53175 Bonn,
ulrike.korte@bsi.bund.de

Abstract: Der Einsatz von qualifizierten elektronischen Signaturen zur elektronischen Übermittlung von Rechnungen gemäß § 14 Abs. 3 [UStG] gewinnt in der Praxis zunehmend an Bedeutung. Da die technischen Formate für qualifizierte elektronische Signaturen nicht gesetzlich vorgeschrieben sind und für den Austauch der Rechnungsdaten verschiedene Datenformate und Übertragungswege genutzt werden, kommen bei der elektronischen Übermittlung von Rechnungen verschiedenste Signaturformate zum Einsatz. Dieser Beitrag liefert einen Überblick über die wichtigsten Signaturformate und gibt Empfehlungen welches Signaturformat in welchem Fall sinnvollerweise eingesetzt werden sollte.

1 Einleitung

Da eine elektronische Rechnung gemäß § 14 Abs. 3 [UStG] mit einer qualifizierten elektronischen Signatur gemäß § 2 Nr. 3 SigG [SigG] versehen sein muss, entwickelt sich die elektronische Übermittlung von Rechnungen in der Praxis zunehmend zu einem besonders wichtigen Einsatzgebiet der qualifizierten elektronischen Signatur. Anders als beispielsweise in Italien [I-MfIT04] schreibt der Deutsche Gesetzgeber aber nicht vor, welche technischen Formate für qualifizierte elektronische Siganturen einzusetzen sind. Deshalb kann für die Übermittlung von Rechnungen grundsätzlich jedes Signaturformat eingesetzt werden, das unter Einsatz von sicheren Signaturerstellungseinheiten gemäß § 2 Nr. 10 [SigG] erzeugt werden kann sofern für den zugehörigen Signaturprüfschlüssel ein qualifiziertes Zertifikat existiert. Da für den Austauch von Rechnungsdaten verschiedene Datenformate und Übertragungswege genutzt werden, kommen bei der elektronischen Übermittlung von Rechnungen in der Praxis auch verschiedenste Signaturformate zum Einsatz.

Dieser Beitrag liefert einen Überblick über die wichtigsten Signaturformate im Umfeld der elektronischen Rechnung und ist folgendermaßen gegliedert: Abschnitt 2 stellt die wichtigsten Aspekte der verschiedenen Signaturformate vor. Nach der ausführlichen Diskussion der generischen, aus dem PKCS #7 Standard hervorgegangenen, Cryptographic Message Syntax (CMS) in Abschnitt 2.1 wird in Abschnitt 2.2 das darauf aufbauende

S/MIME-Format für die Signatur von E-Mails erläutert. Das für die Signatur von XML-basierten Daten eingesetzte XML Digital Signatur Format wird in Abschnitt 2.3 erläutert. Der Standard für die Signatur von EDIFACT-Daten wird in Abschnitt 2.4 vorgestellt. In Abschnitt 2.5 wird darauf eingegangen, wie PKCS #7 Signaturen in PDF-Dokumente integriert werden können.

Abschnitt 3 enthält schließlich einfache Empfehlungen, in welchen Situationen welche Signaturformate sinnvollerweise eingesetzt werden sollten.

2 Signaturformate

Bei Signaturformaten kann in grober Art und Weise zwischen grundlegenden Low-Level-Signaturformaten und erweiterten High-Level-Signaturformaten unterschieden werden. Bei Low-Level-Signaturformaten ist spezifiziert, wie die zu signierenden Daten, oder ein Hashwert derselben, vor der Anwendung des eigentlichen Signaturalgorithmus, z.B. durch Padding-Mechanismen, aufzubereiten sind. Das Format selbst ist maßgeblich vom eingesetzten Kryptoalgorithmus (z.B. RSA, DSA, ECDSA) geprägt.

High-Level-Signaturformate umfassen die rohen Signaturdaten, wie sie durch das Low-Level-Signaturformat vorgegeben sind, sowie weitere Informationen, die bei der Gültigkeitsprüfung der Signatur herangezogen werden, wie z.B. den Zeitpunkt der Signaturerstellung und das zur Prüfung der Signatur notwendige Zertifikat.

Während anhand einer Signatur im Low-Level-Format nur die mathematische Gültigkeit der Signatur geprüft werden kann, unterstützen die High-Level-Signaturformate in der Regel die komplette Gültigkeitsprüfung von Signaturen und Zertifikaten sowie beispielsweise auch die Mehrfachsignatur.

Für den Einsatz zur Signatur von elektronischen Rechnungen ist die ausschließliche Verwendung von Low-Level-Signaturformaten wenig geeignet, da sonst die für die Prüfung der qualifizierten elektronischen Signatur zusätzlich notwendigen Daten (Signaturerstellungszeitpunkt, Zertifikat, Sperrinformationen, weitere Attribute etc.) in einer nicht standardisierten Art und Weise an den Empfänger der Rechnung übertragen werden müssten.

Deshalb werden im Folgenden die wichtigsten High-Level-Signaturformate für die elektronische Übermittlung von Rechnungen vorgestellt.

2.1 Cryptographic Message Syntax / PKCS #7

Mit der auf den PKCS #7 - Standard [PKCS7(v1.5)] zurückgehenden Cryptographic Message Syntax (CMS) [RFC2630, RFC3369, RFC3852] wurde das in der Praxis vielleicht gebräuchlichste Format für kryptographisch behandelte Nachrichten spezifiziert.

Die Betrachtung des CMS-Standards ist in folgende Abschnitte gegliedert:

- Containertypen für CMS

- Die Struktur von `SignedData` und `SignerInfo`
- Abläufe bei der Erstellung und Prüfung von Signaturen
- Enveloping und Detached Signaturen mit CMS

2.1.1 Containertypen für CMS

Die CMS-Spezifikation sieht die Ablage von Nachrichten in `ContentInfo`-Containern vor. Diese Container haben jeweils einen bestimmten Typ, wobei in [RFC3852] folgende Containertypen definiert sind:

- `Data`
  bezeichnet beliebige Daten.
- `SignedData`
  wird für elektronische Signaturen verwendet und im folgenden näher erläutert.
- `EnvelopedData`
  enthält mit einem Nachrichtenschlüssel verschlüsselte Daten und die Verschlüsselung des verwendeten Nachrichtenschlüssels für den oder die Empfänger.
- `DigestedData`
  enthält Daten, die zum Schutz der Integrität um einen Hashwert der Daten ergänzt wurden.
- `EncryptedData`
  enthält verschlüsselte Daten. Anders als beim `EnvelopedData`-Typ wird hier allerdings der zur Verschlüsselung verwendete Nachrichtenschlüssel nicht zusätzlich für die Empfänger verschlüsselt. Das Schlüsselmanagement muss also auf einem anderen Weg geschehen.
- `AuthenticatedData`
  enthält Daten, die mit einem Message Authentication Code (MAC) gesichert sind, sowie die Verschlüsselung des zur MAC-Erzeugung verwendeten symmetrischen Schlüssels für einen oder mehrere Empfänger.
- Weitere Containertypen
  können auch außerhalb der CMS-Spezifikation [RFC3852] definiert werden. Insbesondere sind hier folgende zu nennen:

– `SignedAndEnvelopedData`
wurde in [PKCS7(v1.5), RFC2315] definiert und erlaubt es, Inhaltsdaten zu signieren und direkt danach zu verschlüsseln. Da man ein ähnliches Ergebnis auch durch die Schachtelung von `SignedData`- und `EnvelopedData`-Containern erreichen kann, wurde dieses Format nicht in die CMS-Spezifikationen der IETF [RFC2630, RFC3369, RFC3852] aufgenommen.

– `CompressedData`
wurde in [RFC3274] definiert und erlaubt die Komprimierung von Daten mit dem ZLIB-Verfahren [RFC1950, RFC1951].

2.1.2 Die Struktur von `SignedData` und `SignerInfo`

Das Herzstück des CMS-Signaturformats ist durch die ASN.1-Strukturen `SignedData` und `SignerInfo` gegeben.

Der `SignedData`-Container enthält im Wesentlichen die zu signierenden Daten und eine oder mehrere Signaturen in Form der `SignerInfo`-Struktur.

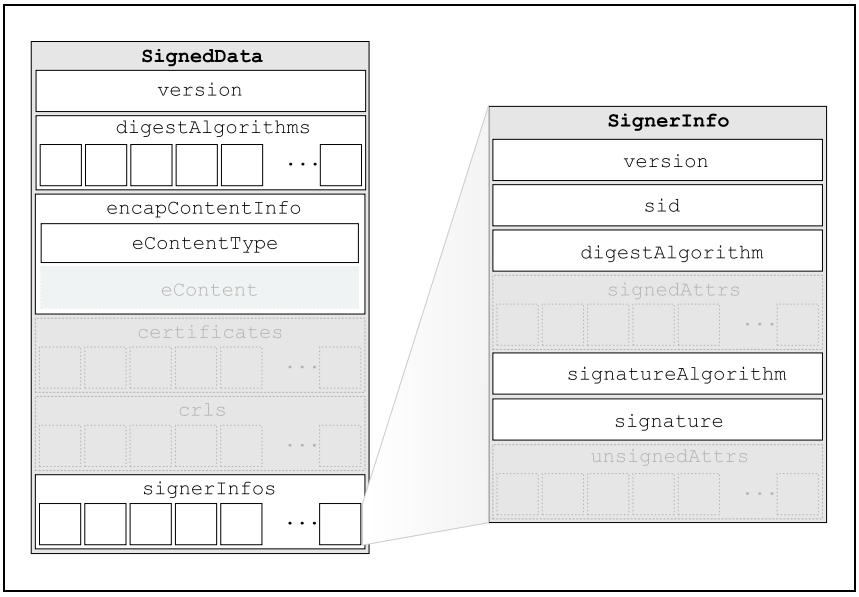

Abbildung 1: Struktur des `SignedData`-Containers aus CMS / PKCS #7

Wie in Abbildung 1 angedeutet, besteht der `SignedData`-Container aus folgenden Teilen:

- `version`
ist ein ganzzahliger Wert, durch den die Kompatibilität zwischen den verschiedenen

Versionen der CMS-Spezifikationen [PKCS7(v1.5), RFC2630, RFC3369, RFC3852] sichergestellt wird.

- `digestAlgorithms`
  listet die Hashalgorithmen auf, die später in den `signerInfos` wieder vorkommen. Durch die Angabe der verwendeten Hashalgorithmen vor dem zu hashenden Inhalt kann eine Verarbeitung in einem Durchlauf erreicht werden.

- `encapContentInfo`
  enthält die Nutzdaten, die signiert werden sollen. Da das `eContent`-Feld in der `encapContentInfo`-Struktur fehlen darf, können auch so genannte „Detached Signatures“ erzeugt werden, die sich auf extern abgelegte Inhalte beziehen (vgl. Abschnitt 2.1.4).

- `certificates`
  ist ein optionales Feld, das eine Menge von Public-Key- und Attributzertifikaten enthalten kann, die zur Signaturprüfung herangezogen werden können.

- `crls`
  ist ein optionales Feld, das Informationen zur Prüfung des Zertifikatsstatus enthalten kann. Gemäß der jüngsten CMS-Spezifikation [RFC3852] können hier neben Sperrlisten auch andere Sperrinformationen, beispielsweise vom Typ `OCSPResponse`, enthalten sein.

- `signerinfos`
  enthält eine Menge von Signaturen, die aus folgenden Informationen zusammengesetzt sind:

  - `version`
    ist ein ganzzahliger Wert, durch den die verschiedenen Versionen der Syntax unterschieden werden können. Unterschiede existieren hier lediglich beim `sid`-Element vom Typ `SignerIdentifier`.
  - `sid`
    ermöglicht die Identifikation des (Zertifikates des) Signierenden.
  - `digestAlgorithm`
    gibt an, welcher Hashalgorithmus zur Erzeugung der Signatur verwendet wurde.
  - `signedAttrs`
    kann eine Menge von Attributen enthalten, die in die Signatur einbezogen werden.

- `signatureAlgorithm`
  enthält den Signaturalgorithmus. Beispielsweise kann hier der in [RFC2313] spezifizierte Algorithmus `rsaEncryption` eingesetzt werden.
- `signature`
  enthält die tatsächliche Signatur im angegebenen Low-Level-Signaturformat.
- `unsignedAttrs`
  kann eine Menge von Attributen enthalten, die jedoch – anders als bei `signedAttrs` oben – nicht in die Signatur einbezogen werden.

Durch die Präsenz mehrerer `SignerInfo`-Felder in einem `SignedData`-Container können *parallele* Mehrfachsignaturen realisiert werden.

Da die zu signierenden Nutzdaten wiederum aus einem `ContentInfo`-Container vom Typ `SignedData` bestehen können, ist es möglich *sequentielle* Mehrfachsignaturen zu konstruieren.

Soll nicht die gesamte `SignedData`-Struktur – und damit möglicherweise mehrere parallele Signaturen – erneut signiert werden, sondern lediglich eine einzelne Signatur „gegengezeichnet“ werden, so kann dies unter Verwendung des `Countersignature`-Attributes geschehen.

2.1.3 Abläufe bei der Erzeugung und Verifikation von Signaturen

Die Erzeugung einer Signatur erfolgt durch die Erzeugung des Hashwertes, ggf. das Padding des Hashwertes und schließlich die Berechnung der Signatur. Während die beiden letzten Schritte vom verwendeten kryptographischen Signaturalgorithmus abhängen, werden die zur Erzeugung des Hashwertes zu verwendenden Daten von der CMS-Spezifikation festgelegt.

Sind keine signierten Attribute vorhanden, so wird einfach der Hashwert des Inhalts[1] des `eContent`-Felds gehasht.

Sind jedoch signierte Attribute vorhanden, so wird das gesamte `signedAttributes`-Feld – die komplette ASN.1/DER-Codierung – für die Berechnung des Hashwertes herangezogen. Da in diesem Fall mindestens auch die beiden Attribute `ContentType` und `MessageDigest` vorhanden sein müssen, fließt der Hashwert des `eContent` indirekt in die Berechnung des Hashwertes ein.

Bei der Prüfung der Signatur verfährt man analog, wobei außerdem die Prüfung des Zertifikatspfades zu erfolgen hat.

2.1.4 Enveloping und Detached Signatures mit CMS

Wie in Abbildung 2 angedeutet, ermöglicht das CMS-Signaturformat zwei grundsätzliche Arten der Signatur:

[1] Der ASN.1-Overhead (`TAG` und `LENGTH`) fließt nicht in die Berechnung des Hashwertes ein.

- Enveloping Signature

  Ist die Nachricht ein Teil der Signatur, so spricht man von einer „Enveloping Signature“. In diesem Fall wird die Nachricht gemäß ASN.1 codiert und in das `eContent`-Feld der `SignedData`-Struktur eingefügt.

- Detached Signature

  Andernfalls fehlt das `eContent`-Feld und die Nachricht wird in einer separaten Datei abgelegt. In diesem Fall spricht man von einer „Detached Signature“. Optional kann der Dateiname der Nachrichtendatei als signiertes Attribut in der Signatur vorhanden sein, um die Zuordnung von Nachricht und Signatur zu erleichtern.

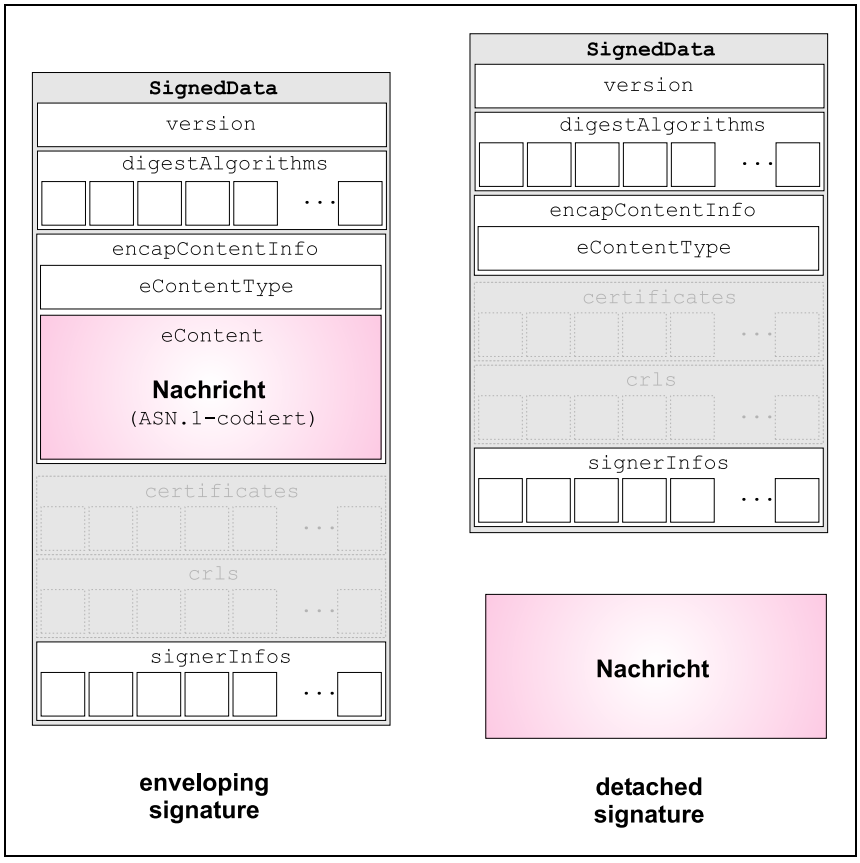

Abbildung 2: Enveloping und Detached Signatures mit CMS

Im ersten Fall liegen die Nachricht und die Signatur in einer einzigen (ASN.1-codierten) Datei vor, so dass keine Probleme mit dem Zusammenfügen auftreten können. Allerdings ist die Nachricht nur lesbar, wenn ein CMS-fähiges Anwendungsprogramm existiert. Da

dies in vielen Fällen problematisch ist, verwendet man in der Praxis häufig die zweite Variante, bei der die Nachricht und die Signatur in separaten Dateien vorliegen. Um die automatisierte Zuordnung einer Nachrichten- und Signaturdatei zu erleichtern, wählt man für die Signaturdatei in der Regel den gleichen Dateinamen und hängt lediglich die zusätzliche Dateiendung .p7s an – standardisiert ist diese Vorgehensweise zur Zuordnung zwischen Signatur- und Nachrichtendatei aber leider nicht.

2.2 S/MIME

Secure Multipurpose Internet Mail Extensions (S/MIME) ist ein ursprünglich von den Laboratorien der RSA Security Inc. entwickeltes, nunmehr in der IETF in [RFC2632, RFC2633] standardisiertes, auf PKCS #7/CMS aufbauendes, Format für die Verschlüsselung und Signatur von E-Mails und E-Mail-Anhängen im MIME-Format [RFC1521].

Für die Signatur von E-Mails sieht der S/MIME-Standard [RFC2633] zwei verschiedene Varianten vor:

- `application/pkcs7-mime` mit `SignedData`
- multipart/signed

Da die signierte Mail bei der multipart/signed-Variante auch mit E-Mail-Clients gelesen werden kann, die S/MIME nicht unterstützen, sollte sie vom Sender generell bevorzugt werden.

2.2.1 `application/pkcs7-mime` mit `SignedData`

In diesem Fall besteht die übertragene Nachricht aus einer CMS-Struktur vom Typ `SignedData`, die in einer MIME-Nachricht vom Typ `application/pkcs7-mime` transportiert wird.

Hierfür wird in einem ersten Schritt die Nachricht in eine kanonische Form gebracht, indem bei einer Text-Nachricht beispielsweise sichergestellt wird, dass eine Zeile mit den Zeichen (CR)/(LF) (Carriage Return und Line Feed) endet. Diese Daten sollten[2] in eine Transport-Codierung, wie base64 oder quoted-printable, überführt werden, so dass eine fehlerfreie Übertragung der E-Mail auch dann gewährleistet ist, wenn eine Mail-Relay-Station auf dem Weg zwischen Sender und Empfänger nur in der Lage ist 7-Bit-Daten zu verarbeiten. Aus diesen Daten, die den `eContent` der CMS-Struktur vom Typ `Data` bilden, wird eine CMS-Signatur gebildet (vgl. Abschnitt 2.1). Diese CMS-Struktur wird schließlich in einen MIME-Typ vom Typ `application/pkcs7-mime` mit `signed-data` im optionalen „smime-type“-Parameter eingebettet.

[2] Auch wenn die Transport-Codierung nur bei der multipart/signed-Nachricht zwingend notwendig ist, wird in [RFC2633, Section 3.1.2] empfohlen, sie bei allen S/MIME-Varianten zu verwenden, da so eine fehlerfreie Übertragung auch dann sichergestellt ist, wenn bei der internen Verarbeitung der E-Mail beim Empfänger Systeme eingesetzt werden, die nur in der Lage sind 7-Bit-Daten zu verarbeiten.

2.2.2 multipart/signed

Bei der multipart/signed-Variante, die auf [RFC1847] zurück geht, besteht die S/MIME-Nachricht aus zwei Teilen, die als MIME-Nachrichten codiert sind. Im ersten Teil befindet sich die zu signierende Nachricht, im zweiten Teil befindet sich die zugehörige Signatur im CMS-Format, in der der `eContent` leer ist. Auch hier wird die zu signierende MIME-Nachricht in eine kanonische Form gebracht und die beiden MIME-Nachrichten in eine entsprechende Transport-Codierung überführt. Der Vorteil dieser Variante ist, dass die signierte Nachricht auch mit nicht-S/MIME-fähigen Mail-Programmen gelesen werden können.

2.3 XML Digital Signature

Für die digitale Signatur von Daten im XML-Format wurde von einer Arbeitsgruppe des W3C ein spezifisches Signaturformat entwickelt [XML-DSig, RFC3275]. Im Vergleich zum in Abschnitt 2.1 erläuterten Cryptographic Message Syntax-Signaturformat bietet diese XML-Signatur ein höheres Maß an Flexibilität, das notwendig ist, um das volle Potenzial von XML auch im Bereich der digitalen Signatur ausnutzen zu können.

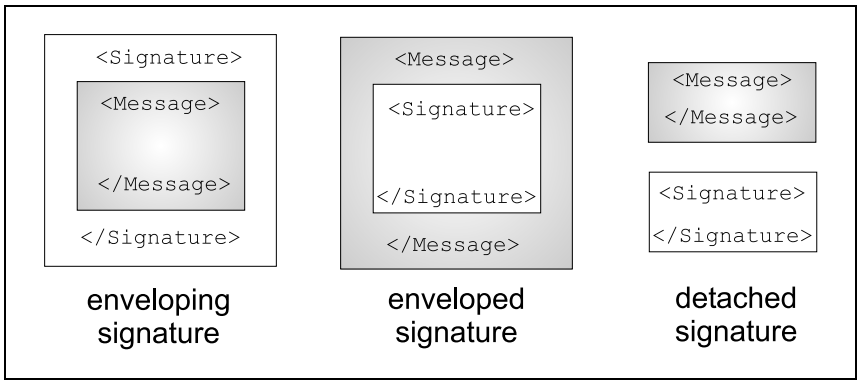

Abbildung 3: XML-Signatur-Typen

Während das CMS-Format nur die Erstellung von umschließenden Signaturen (enveloping signature) und von der Nachricht getrennten Signaturen (detached signature) unterstützt, kann bei der XML-Signatur gemäß [XML-DSig, RFC3275] auch die Signatur in die zu signierende Nachricht eingebettet sein (enveloped signature) (vgl. Abbildung 3). Ein ähnliches Konstrukt ist für CMS-basierte Signaturen nur in Verbindung mit bestimmten Dokumenten-Typen, wie beispielsweise PDF (vgl. Abschnitt 2.5), in standardisierter Art und Weise möglich.

Bei der XML-Signatur kann man komplette Dateien eines beliebigen Typs oder ganz gezielt bestimmte Teile eines XML-Dokumentes in die Signatur einbeziehen oder die zu

```
<Signature ID>
     <SignedInfo>
          <CanonicalizationMethod/>
          <SignatureMethod/>
          <Reference URI>
                  <Transforms>
                  <DigestMethod>
                  <DigestValue>
          </Reference>
     </SignedInfo>
     <SignatureValue>
     <KeyInfo>
     <Object ID>
</Signature>
```

Abbildung 4: Struktur einer XML-Signatur

signierenden Daten vor der Signaturerstellung in einer bestimmten Art und Weise transformieren. Hierbei können beispielsweise XPath- oder XSL-Transformationen durchgeführt werden. Mit der XPath-Transformation [XPath] ist es möglich bestimmte Bestandteile eines XML-Dokumentes bei der Erzeugung der Signatur auszusparen, so dass sich diese Datenfelder später ändern können, ohne dass die Signatur ungültig werden würde. Dies kann[3] beispielsweise für die Erzeugung von eingebetteten Signaturen (enveloped signature) verwendet werden. Mit einer XSL-Transformation [XSLT] können die Daten im XML-Format vor der Erzeugung und Prüfung einer Signatur mit einem bestimmten Layout [XSL] verknüpft werden.

Eine XML-Signatur beginnt, wie in Abbildung 4 dargestellt, mit dem `<Signature>`-Tag, endet mit dem `</Signature>`-Tag und enthält im Wesentlichen folgende Bestandteile:

- `SignedInfo`

  Enthält Informationen wie, welche Daten zu signieren sind. Insbesondere enthält es folgende Bestandteile:

  - `CanonicalizationMethod`

    gibt an, welche Kanonisierungsmethode eingesetzt wird.

  - `SignatureMethod`

    spezifiziert den Signaturalgorithmus.

  - `Reference`

    ist ein Feld, das ein- oder mehrmals vorhanden sein kann und einen Verweis auf die zu signierenden Daten, die mittels eines Uniform Resource Identifier (URI) adressiert werden, und Informationen zur Aufbereitung derselben enthält. Es enthält insbesondere folgende Elemente:

[3] Für den Fall der eingebetteten Signatur existiert außerdem eine spezielle Transformation (vgl. [RFC3275, Section 6.6.4]).

    - `Transforms`
      gibt möglicherweise an, wie die Daten vor der Anwendung der Hashfunktion aufzubereiten sind.
    - `DigestMethod`
      spezifiziert die zu verwendende Hashfunktion.
    - `DigestValue`
      enthält den Hashwert, der mit der angegebenen Hashfunktion aus den referenzierten und möglicherweise transformierten Daten berechnet wurde.

- `SignatureValue`
  enthält die Signatur.

- `KeyInfo`
  ist ein optionales Feld, in dem beispielsweise Zertifikate abgelegt sein können.

- `Object`
  kann beliebig oft (auch keinmal) auftreten und ein beliebiges Datenobjekt beinhalten. Beispielsweise kann hier eine zu signierende Nachricht, zusätzliche Eigenschaften der Signatur (`SignatureProperties`), eine weitere Signatur oder ein so genanntes `Manifest` in die Signatur einbezogen werden. Ein `Manifest` enthält eine Liste von Referenzen auf bestimmte Daten und den zugehörigen Hashwert derselben.

Anders als beim CMS-Signaturformat werden die Daten nicht gemäß ASN.1 codiert, sondern liegen, wie in Abbildung 4 angedeutet, in „lesbaren“ XML-Strukturen vor.

2.4 EDIFACT

EDIFACT ist ein branchenübergreifender internationaler Standard für den automatisierten Austausch elektronischer Daten im Geschäftsverkehr, dessen Syntax in ISO9735 festgelegt ist.

Wie in Abbildung 5 angedeutet, besteht eine EDIFACT-Übertragungsdatei (Interchange) aus einem optionalen UNA-Segment, durch das u.a. die verwendeten Trennzeichen festgelegt sind, dem Interchange-Header-Segment (UNB), den zu übertragenden Daten und schließlich dem Interchange-Trailer-Segment (UNZ). Die zu übertragenden Daten bestehen entweder aus Gruppen (von Nachrichten oder Paketen) oder nur aus Nachrichten oder Paketen. Eine Nachricht besteht wiederum aus einem Nachrichtenkopf (Message-Header, UNH), möglicherweise[4] einer Security Header Gruppe, dem Nachrichtenrumpf (Message Body), einer Security Trailer Gruppe und schließlich dem Nachrichten-Ende-Segment (Message-Trailer, UNT). Seit der Version 4 des ISO9735-Standards gibt es auch

[4] Die Verwendung von Sicherheitsmechanismen für EDIFACT ist optional und erst mit der Syntax Version 4 eingeführt worden.

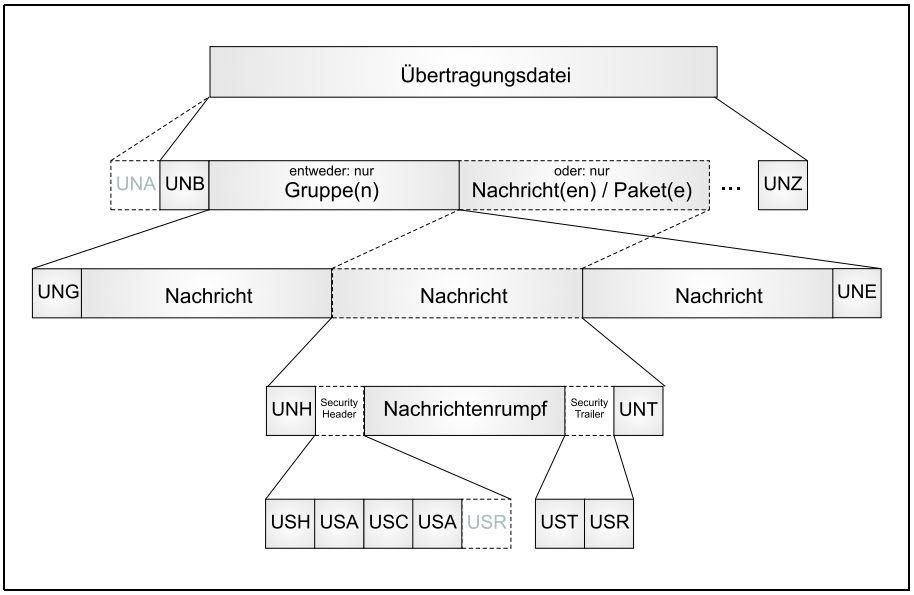

Abbildung 5: Beispielhafte EDIFACT-Übertragungsdatei

die Möglichkeit Paket-Objekte einzubinden. Hierfür wird der Objekt-Header (UNO) und der Objekt-Trailer (UNP) eingesetzt.

Wie in [ISO9735-5] spezifiziert, besteht die Security Header Gruppe aus folgenden Segmenten:

- Security Header (USH)

  gibt an, welcher Sicherheitsdienst, in welcher Art und Weise, auf die eingeschlossenen oder referenzierten Daten angewandt wird. Beispielsweise wird man beim Einsatz digitaler Signaturen zur Übermittlung elektronischer Rechnungen in Form von EDIFACT [INVOIC]-Nachrichten hier den Sicherheitsdienst „Non-repudiation of origin“ angeben. Neben dem Sicherheitsdienst können weitere Elemente, wie z.B. eine Sicherheitsfolgenummer oder ein Zeitstempel, angegeben werden.

- Security Algorithm (USA)

  spezifiziert im allgemeinen den zu verwendenden kryptographischen Algorithmus. Beim Einsatz der digitalen Signatur wird hier der verwendete Hashalgorithmus festgelegt.

- Certificate (USC)

  enthält einen öffentlichen Schlüssel oder Informationen zu einem verwendeten Zertifikat, wie beispielsweise die Seriennummer und den Aussteller des Zertifikates.

- Security Algorithm (USA)

  Bei der digitalen Signatur kann dieses USA-Segment in drei Ausprägungen vorkommen:

  a) als der vom Aussteller eines EDIFACT-Zertifikates verwendete Algorithmus zur Berechnung des Hashwertes des Zertifikats,
  b) als der vom Zertifikatsaussteller verwendete Signaturalgorithmus zur Erstellung des Zertifikats und
  c) als der vom Absender verwendeten Signaturalgorithmus zur Signierung einer Nachricht bzw. eines Pakets.

  Sofern X.509-Zertifikate eingesetzt werden, enthält dieses USA-Segment also den zur Signatur der Nutzdaten eingesetzten Algorithmus.

- Security Result (USR)

  enthält möglicherweise die Signatur des im USC-Segment angegebenen öffentlichen Schlüssels. Wird im USC-Segment auf ein X.509-Zertifikat verwiesen, so kann dieses USR-Segment entfallen.

Die Security Trailer Gruppe besteht aus folgenden Segmenten:

- Security Trailer (UST)

  sorgt für die Verbindung zur oben erläuterten Security Header Gruppe und gibt an, aus wie vielen USR-Segmenten die Security Trailer Gruppe besteht.

- Security Result (USR)

  enthält schließlich das Ergebnis der Sicherheitsoperation, beispielsweise eine digitale Signatur.

Neben der Signatur von EDIFACT-Nachrichten mittels Header und Trailer kann auch die in [ISO9735-6] spezifizierte AUTACK-Nachricht zum Einsatz kommen. Wie in Abbildung 6 dargestellt, dient die AUTACK entweder der *AUT*hentisierung von gesendeten Übertragungsdateien, Gruppen, Nachrichten oder Paketen oder der kryptographisch gesicherten Bestätigung des Empfangs (*ACK*nowledgement) von Übertragungsdateien, Gruppen, Nachrichten oder Paketen.

Die AUTACK enthält selbst eine oben erläuterte Security Header (USH) Gruppe und ein Security Trailer Segment (UST), die eine Reihe von AUTACK-spezifischen Segmenten umfassen:

- Secured Data Identification (USB)

  enthält Informationen zum Sender und Empfänger der AUTACK-Nachricht.

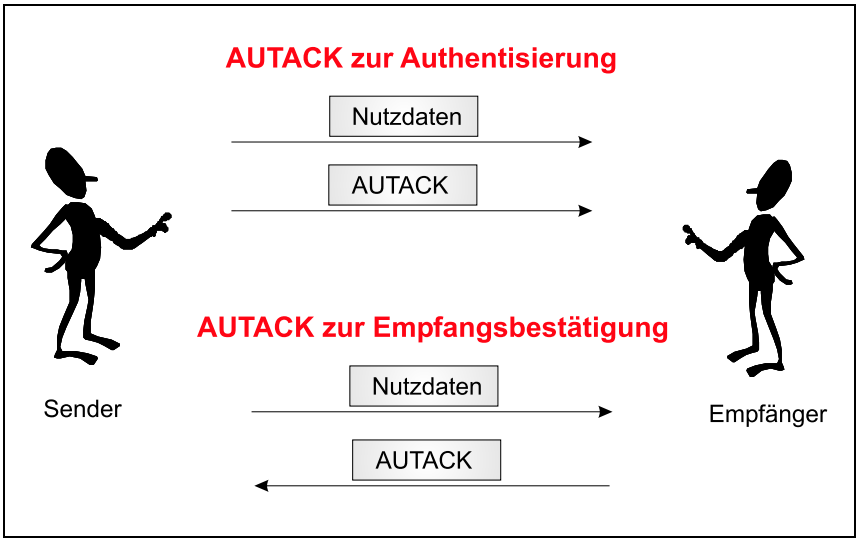

Abbildung 6: Einsatz der AUTACK-Nachricht

- Security References (USX)

  enthält den Verweis auf die durch die AUTACK-Nachricht gesicherten Daten in Form von Referenznummern für Interchanges oder Messages.
- Security on References (USY)

  enthält das Ergebnis der Sicherheitsoperation, die auf die referenzierten Daten angewandt wird. Beispielsweise kann hier eine digitale Signatur der referenzierten Daten enthalten sein.

Eine AUTACK-Authentisierungs-Nachricht kann auf zwei Arten angewendet werden. Die erste Methode transportiert die Hashwerte der referenzierten EDIFACT-Strukturen, die durch die AUTACK selbst gesichert werden. Bei der zweiten Methode wird die AUTACK nur dazu verwendet, die digitalen Signaturen der referenzierten EDIFACT-Strukturen zu transportieren.

Da bei der Verwendung der AUTACK-Nachricht die eigentlichen EDIFACT-Nutzdaten, beispielsweise eine Rechnung im [INVOIC]-Format, noch unter Verwendung der EDIFACT Syntax Version 3 vorliegen können, empfiehlt der EDI-Anwenderkreis Handel in [EDI-AK-Handel] den Einsatz der AUTACK zur Übermittlung elektronischer Rechnungen.

Auch bei den EDIFACT-Signaturen wird, ähnlich wie bei der XML-Signatur oben keine ASN.1-Codierung eingesetzt, sondern auf im EDIFACT-Umfeld übliche Codes für bestimmte Datenstrukturen zurückgegriffen.

2.5 Einbettung von Signaturen in PDF-Dokumente

Wie oben erläutert, ist es bei CMS-Signaturen (vgl. Abschnitt 2.1) nicht in jedem Fall möglich die Signatur in standardisierter Art und Weise in ein signiertes Dokument einzubetten (vgl. „enveloped signature“ in Abbildung 3). Eine Ausnahme bildet das PDF-Format [PDF(v1.3), PDF(v1.4), PDF(v1.5), PDF(v1.6)], das spezifische Felder vorsieht, in denen Signaturen eingefügt werden können. Eine Signatur wird in Form eines Verzeichnisses (Signature Directory) in das PDF-Dokument eingebettet. Dieses Verzeichnis kann beispielsweise aus folgenden Feldern bestehen:

- `Type`
  gibt den Typ des Verzeichnisses an. Der Wert `Sig` gibt an, dass es sich um ein Signature Directory handelt.
- `Filter`
  enthält den Namen des zu verwendenden Signatur-Handlers, z.B. `AdobePPKLite`.
- `SubFilter`
  gibt an, um welche Art der Signatur es sich handelt, z.B. `adbe.pkcs7.detached`.
- `ByteRange`
  gibt an, auf welche Daten sich die Signatur bezieht, wobei ein Teil des Dokuments jeweils durch einen Offset und seine Länge referenziert wird. Wie bei der XML-enveloped-signature muss auch hier der Bereich in den die Signatur eingefügt wird bei der Signaturerzeugung ausgespart werden.
- `Contents`
  enthält die digitale Signatur.

Der Vorteil der Einbettung der Signatur in das PDF-Dokument besteht insbesondere darin, dass lediglich ein einziges Dokument verwaltet werden muss und die Prüfung der Signatur mit einer aktuellen Version des kostenlosen Adobe Reader möglich ist. Da dieser meist zur Standard-Rechnerausstattung zählt, kann für die Prüfung elektronischer Signaturen auf die zusätzliche Installation einer Signaturanwendungskomponente verzichtet werden.

3 Zusammenfassung

In diesem Beitrag wurden die für die elektronische Übermittlung von Rechnungen wichtigsten Signaturformate näher beleuchtet. Abschließend sollen einige einfache Empfehlungen ausgesprochen werden, welche der vorgestellten Signaturformate in welchem eBilling-Szenario sinnvollerweise eingesetzt werden sollten.

1. Strukturierte Daten?

   Werden XML- oder EDIFACT-Daten ausgetauscht, so sollten die dafür vorgesehenen, in Abschnitt 2.3 und Abschnitt 2.4 vorgestellten, Signaturformate eingesetzt werden.

2. Textbasierte Rechnungen per E-Mail?

   Werden die Rechnungsdaten per E-Mail versandt, so können die in Abschnitt 2.2 vorgestellten S/MIME-Signaturen eingesetzt werden. Vorteilhaft ist die Nutzung von S/MIME insbesondere dann, wenn die Rechnung als vollständig textbasiertes Dokument im Body der E-Mail übermittelt werden kann.

3. Rechnung im PDF-Format?

   Damit die elektronische Rechnung ein ähnliches Layout erhält, wie eine papiergebundene Rechnung auf Briefpapier etc. wird man für die elektronische Übermittlung von unstrukturierten Rechnungen häufig das PDF-Format nutzen. In diesem Fall wird man das PKCS #7 Format nutzen und die Signatur, wie in Abschnitt 2.5 erläutert, in das PDF-Dokument integrieren. Dies hat den Vorteil, dass nur eine Datei verwaltet werden muss, PDF-Dateien in aller Regel nicht von E-Mail-Content Filtern blockiert werden und die Prüfung der Signatur direkt mit dem Adobe Reader möglich ist.

4. Bei allen anderen Dokumentenformaten sollte man das in Abschnitt 2.1 ausführlich erläuterte CMS / PKCS #7 - Format einsetzen.

Ein großes Unternehmen, das von seinen Geschäftspartnern Rechnungen auf den verschiedensten Wegen erhalten, werden oft nicht umhin kommen alle vorgestellten Signaturformate verarbeiten zu können.

Literatur

[EDI-AK-Handel] EDI ANWENDERKREIS HANDEL. *Empfehlung des AK-Handel zur digitalen Signatur von EDIFACT-INVOIC-Daten*. http://www.edi-ak-handel.de/download/AKH-AgDISI.pdf, 2004.

[I-MfIT04] MINISTER FÜR INNOVATION UND TECHNOLOGIE (ITALIEN). *Technische Vorschriften für die Anerkennung und Prüfung elektronischer Dokumente*. Verordnungsentwurf zur Notifikation durch die Europäische Union, 20041022, 2004.

[INVOIC] CENTRALE FÜR COORGANISATION (CCG). *Übermittlung von Abrechnungsdaten (Rechnungslistensummen) mit EANCOM INVOIC 008*. http://www.edi-ak-handel.de/ak_handel_en/Guides/reli.pdf, Mai 2000.

[ISO9735-5] JOINT ISO/TC 154 UN/CEFACT SYNTAX WORKING GROUP (JSWG). *Electronic data interchange for administration, commerce and transport (EDIFACT) Application level syntax rules (Syntax version number: 4, Syntax release*

number: 1) Part 5: Security rules for batch EDI (authenticity, integrity and non-repudiation of origin). ISO 9735-5 (Second edition 2002-07-01). http://www.gefeg.com/jswg/v41/data/V41-9735-5.zip, Juli 2002.

[ISO9735-6] JOINT ISO/TC 154 UN/CEFACT SYNTAX WORKING GROUP (JSWG). *Electronic data interchange for administration, commerce and transport (EDIFACT) Application level syntax rules (Syntax version number: 4, Syntax release number: 1) Part 6: Secure authentication and acknowledgement message (message type AUTACK)*. http://www.gefeg.com/jswg/v41/data/V41-9735-6.zip, Juli 2002.

[PDF(v1.3)] ADOBE SYSTEMS INCORPORATED. *PDF Reference – Second Edition – Adobe Portable Document Format Version 1.3*. Addison Wesley, ISBN 0-201-61588-6. http://partners.adobe.com/public/developer/en/pdf/PDFReference13.pdf, Juli 2000.

[PDF(v1.4)] ADOBE SYSTEMS INCORPORATED. *PDF Reference – Third Edition – Adobe Portable Document Format Version 1.4*. Addison-Wesley, ISBN 0-201-75839-3. http://partners.adobe.com/public/developer/en/pdf/PDFReference.pdf, November 2001.

[PDF(v1.5)] ADOBE SYSTEMS INCORPORATED. *PDF Reference – Fourth Edition – Adobe Portable Document Format Version 1.5*. http://partners.adobe.com/public/developer/en/pdf/PDFReference15_v6.pdf, August 2003.

[PDF(v1.6)] ADOBE SYSTEMS INCORPORATED. *PDF Reference – Fifth Edition – Adobe Portable Document Format Version 1.6*. http://partners.adobe.com/public/developer/en/pdf/PDFReference16.pdf, November 2004.

[PKCS7(v1.5)] RSA LABORATORIES. *PKCS #7: Cryptographic Message Syntax Standard - Version 1.5*. Public Key Cryptography Standards – PKCS #7. http://www.rsalabs.com, November 1993.

[RFC1521] N. BORENSTEIN und N. FREED. *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*. Request For Comments – RFC 1521. http://www.ietf.org/rfc/rfc1521.txt, September 1993.

[RFC1847] J. GALVIN, S. MURPHY, S. CROCKER, und N. FREED. *Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted*. Request For Comments – RFC 1847. http://www.ietf.org/rfc/rfc1847.txt, Oktober 1995.

[RFC1950] P. DEUTSCH und J-L. GAILLY. *ZLIB Compressed Data Format Specification version 3.3*. Request For Comments – RFC 1950. http://www.ietf.org/rfc/rfc1950.txt, Mai 1996.

[RFC1951] P. DEUTSCH. *DEFLATE Compressed Data Format Specification version 1.3*. Request For Comments – RFC 1951. http://www.ietf.org/rfc/rfc1951.txt, Mai 1996.

[RFC2313] B. KALISKI. *PKCS #1: RSA Encryption - Version 1.5*. Request For Comments – RFC 2313. http://www.ietf.org/rfc/rfc2313.txt, 1998.

[RFC2315] B. KALISKI. *PKCS #7: Cryptographic Message Syntax - Version 1.5*. Request For Comments – RFC 2315. http://www.ietf.org/rfc/rfc2315.txt, 1998.

[RFC2630] R. HOUSLEY. *Cryptographic Message Syntax (CMS)*. Request For Comments – RFC 2630. http://www.ietf.org/rfc/rfc2630.txt, Juni 1999.

[RFC2632] B. RAMSDELL. *S/MIME Version 3 Certificate Handling*. Request For Comments – RFC 2632. http://www.ietf.org/rfc/rfc2632.txt, Juni 1999.

[RFC2633] B. RAMSDELL. *S/MIME Version 3 Message Specification*. Request For Comments – RFC 2633. http://www.ietf.org/rfc/rfc2633.txt, Juni 1999.

[RFC3274] P. GUTMANN. *Compressed Data Content Type for Cryptographic Message Syntax (CMS)*. Request For Comments – RFC 3274. http://www.ietf.org/rfc/rfc3274.txt, Juni 2002.

[RFC3275] D. EASTLAKE, J. REAGLE, und D. SOLO. *(Extensible Markup Language) XML-Signature Syntax and Processing*. Request For Comments – RFC 3275. http://www.ietf.org/rfc/rfc3275.txt, März 2002.

[RFC3369] R. HOUSLEY. *Cryptographic Message Syntax (CMS)*. Request For Comments – RFC 3369. http://www.ietf.org/rfc/rfc3369.txt, August 2002.

[RFC3852] R. HOUSLEY. *Cryptographic Message Syntax (CMS)*. Request For Comments – RFC 3852. http://www.ietf.org/rfc/rfc3852.txt, Juli 2004.

[SigG] *Gesetz über Rahmenbedingungen für elektronische Signaturen und zur Änderung weiterer Vorschriften, vom 16.05.2001*. BGBl. 2001 Teil I Nr. 22, S. 876 ff, Geändert durch Art. 1 G v. 4. 1.2005 I 2. http://bundesrecht.juris.de/bundesrecht/sigg_2001/, 2001.

[UStG] *Umsatzsteuergesetz*. vom 26. November 1979, BGBl I 1979, 1953, Neugefasst durch Bek. v. 21.2.2005 I 386. http://bundesrecht.juris.de/bundesrecht/ustg_1980/gesamt.pdf, 1979.

[XML-DSig] D. EASTLAKE, J. REAGLE, und D. SOLO. *XML-Signature Syntax and Processing*. W3C Recommendation. http://www.w3.org/TR/xmldsig-core/, Februar 2002.

[XPath] J. CLARK und S. DEROSE. *XML Path Language (XPath) Version 1.0*. W3C Recommendation. http://www.w3.org/TR/1999/REC-xpath-19991116, Oktober 1999.

[XSL] S. ADLER, A. BERGL, J. CARUSO, S. DEACH, P. GROSSO, E. GUTENTAG, A. MILOWSKI, S. PARNELL, J. RICHMAN, und S. ZILLES. *Extensible Stylesheet Language (XSL)*. W3C Proposed Recommendation. http://www.w3.org/TR/2001/PR-xsl-20010828/, August 2001.

[XSLT] J. CLARK. *XSL Transforms (XSLT) Version 1.0*. W3C Recommendation. http://www.w3.org/TR/1999/REC-xslt-19991116.html, November 1999.