DaimlerChrysler

Konzepte und Standards sicherer Web-Services

Mario Jeckle
DaimlerChrysler Forschungszentrum Ulm
mario.jeckle@daimlerchrysler.com
mario@jeckle.de
www.jeckle.de

Gliederung

- Anforderungen an SOAP-Sicherheitsmechanismen
- Existierende Lösungen
  - XML Digital Signatures
  - XML Encryption
  - Secure Sockets Layer
- (Code-)Beispiele und Praktische Einsatzempfehlungen

Anforderungen an einen SOAP-Sicherheitsmechanismus

- Vertraulichkeit (confidentiality)

  Schutz der Daten vor dem (lesenden) Zugriff unbefugter Dritter

- Berechtigung (authorization)

  Gewährleistet Befugnis des Anforderers zur Nutzung des Dienstes

- (Daten-)konsistenz (data integrity)

  Verlangt modifikationsfreies Eintreffen der versandten Daten

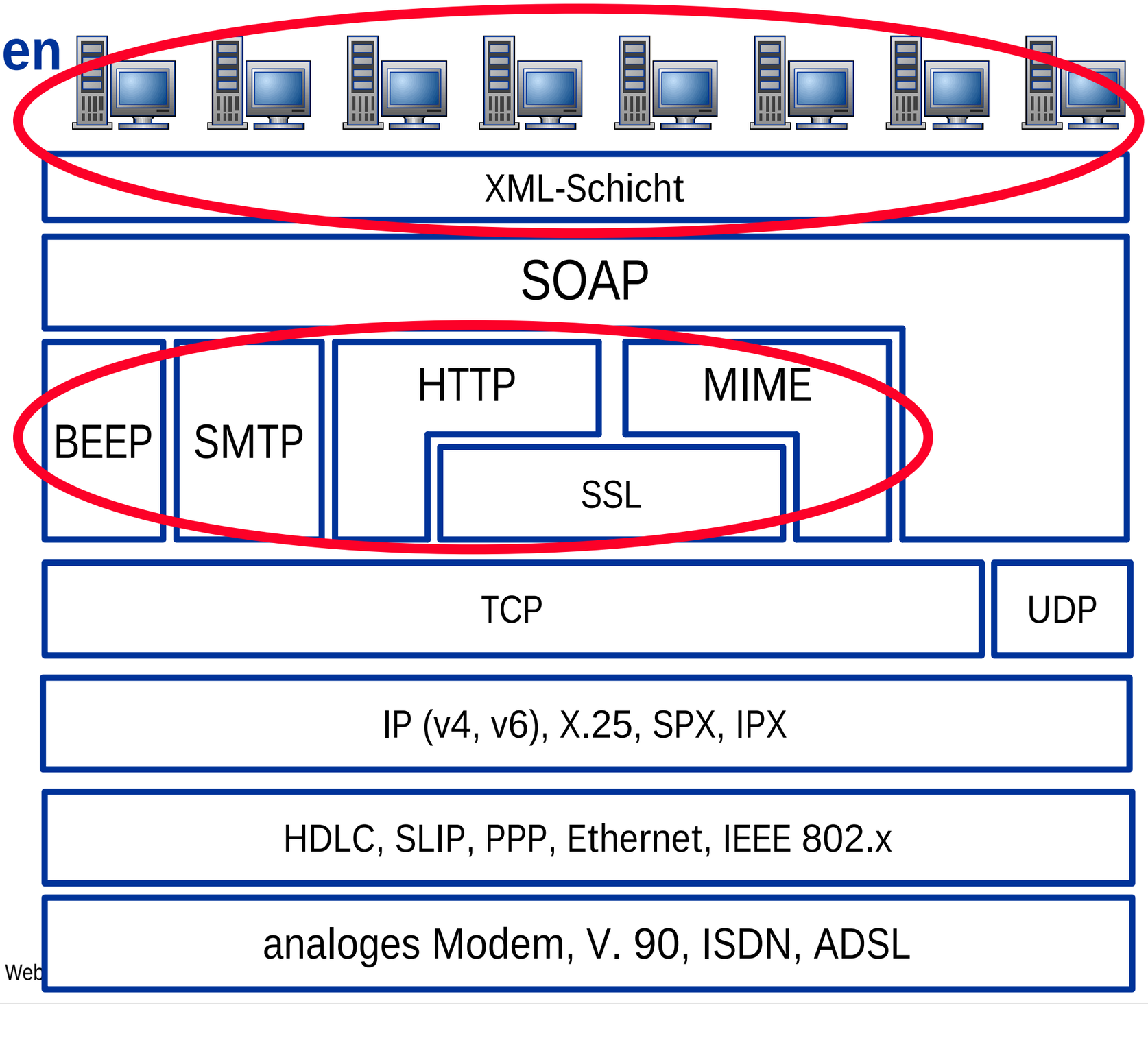
- Glaubwürdigkeit des Ursprungs (message origin authentication)

  Garantiert, daß eine Nachricht willentlich durch einen Sender erstellt wurde

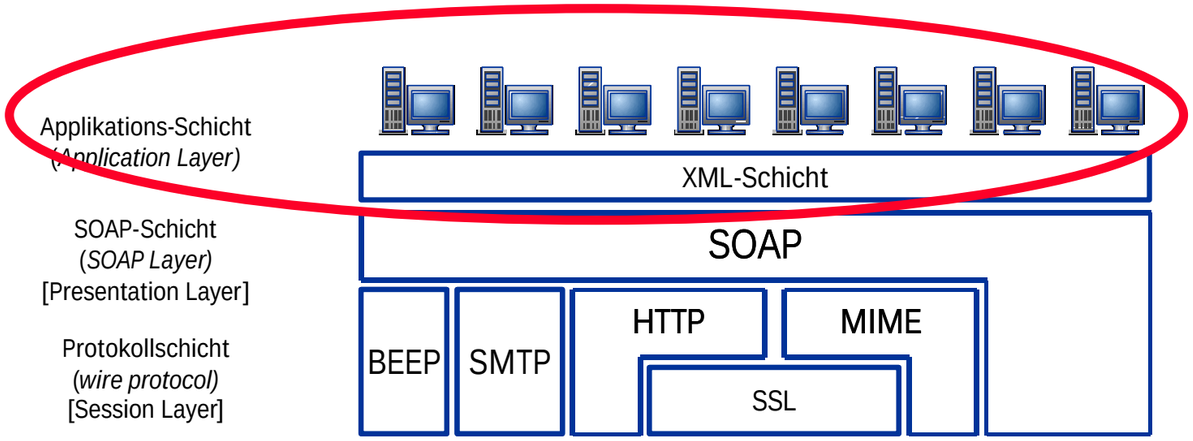
- Verbindlichkeit (non-repudiation)

  Stellt sicher, daß der Sender die Autorenschaft nicht leugnen kann

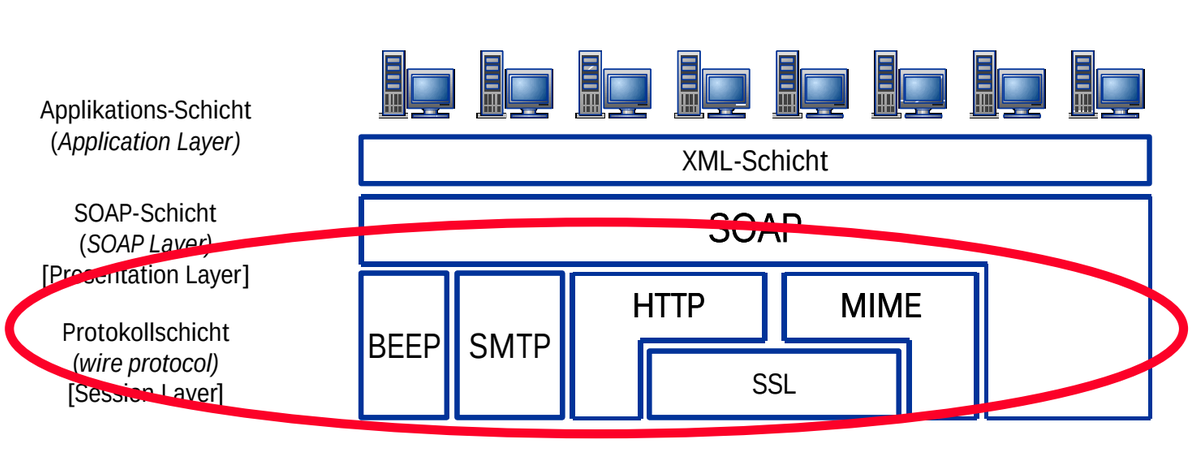
Sicherheitsebenen



Sicherheitsebenen

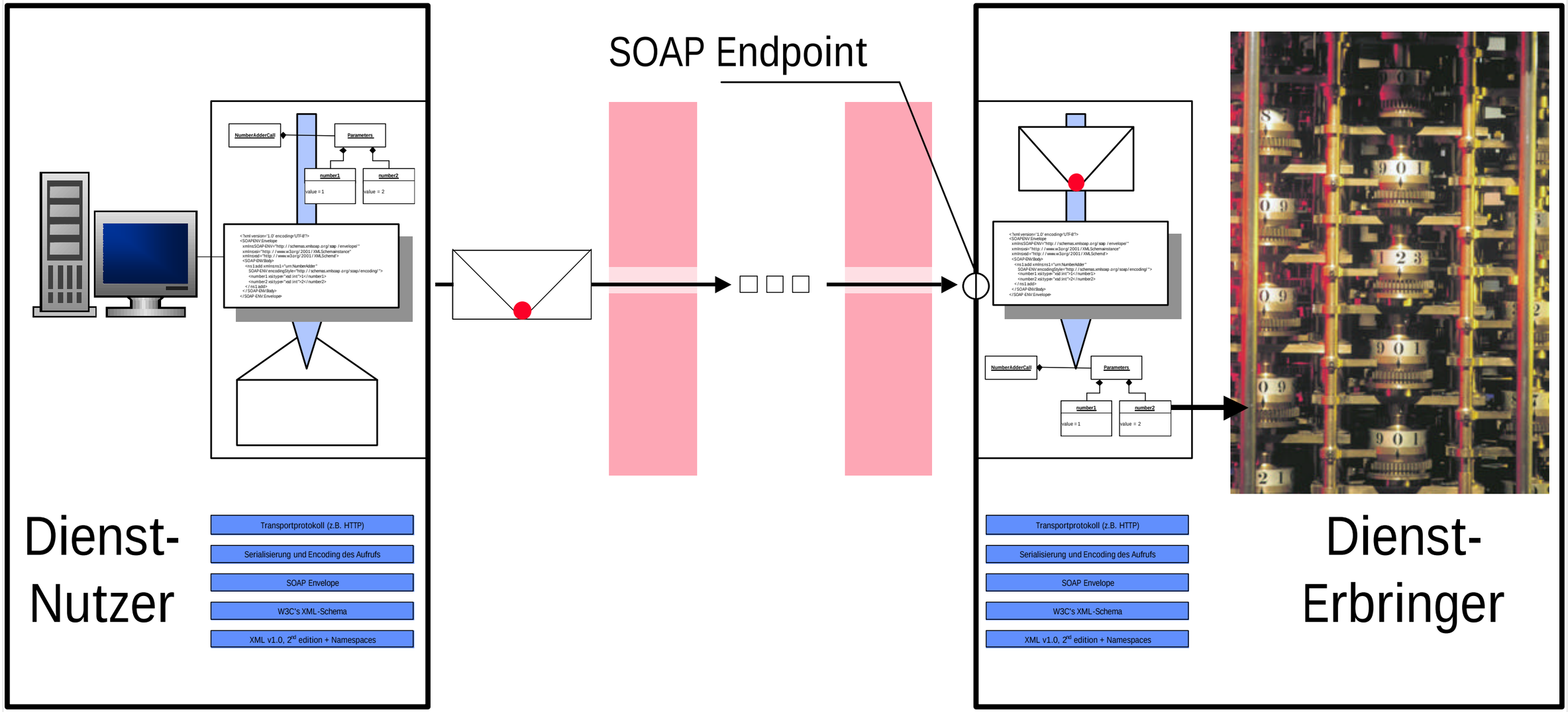


- XML Digital Signatures
- XML Encryption



- Transaction Layer Security / Secure Sockets Layer

Ein Beispiel ...



Dienstaufruf (ungesichert)

```
POST /axis/Calculator.jws HTTP/1.0
Content-Length: 478
Host: 53.16.71.32
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://www.daimlerchrysler.com/add"

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <SOAP-ENV:Body>
  <ns1:add xmlns:ns1="http://www.daimlerchrysler.com/">
   <op1 xsi:type="xsd:int">2</op1>
   <op2 xsi:type="xsd:int">4</op2>
  </ns1:add>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Dienstergebnis (ungesichert)

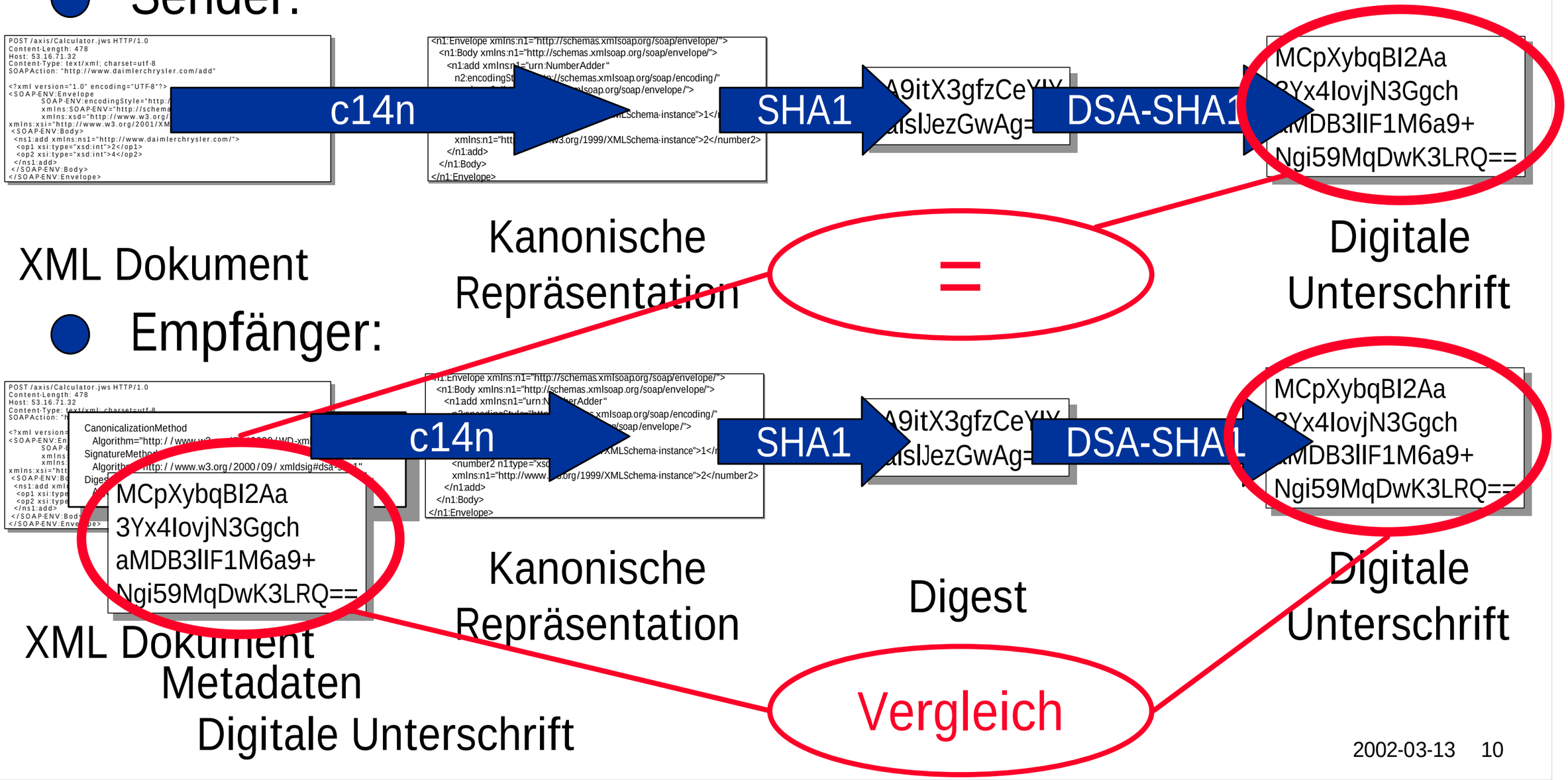
```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 419
Date: Wed, 13 Mar 2002 12:41:32 GMT
Server: Apache Tomcat/4.0.3 (HTTP/1.1 Connector)

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <SOAP-ENV:Body>
  <ns1:addResponse xmlns:ns1="http://www.daimlerchrysler.com/">
   <addResult xsi:type="xsd:int">6</addResult>
  </ns1:addResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

XML Digital Signatures -- Intention

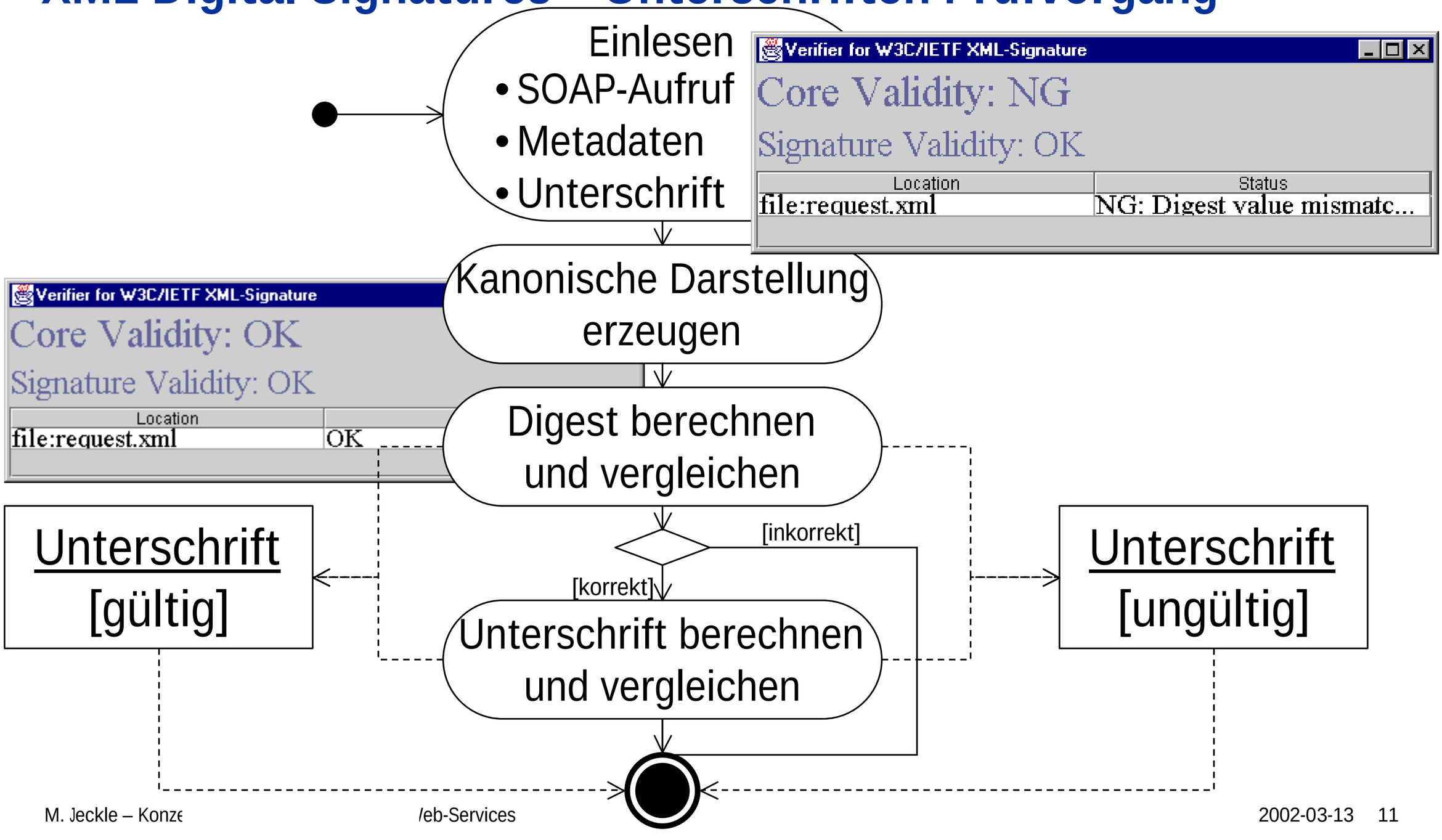
- Ziel: Aufdeckung potentieller Datenverfälschung
- Sender „unterschreibt“ übertragene Daten,
  Empfänger prüft Unterschrift (und mittels dieser indirekt die Daten)
- Eigenschaften der Unterschrift:
  - Glaubwürdigkeit (willentliche Unterschrift)
  - Fälschungssicherheit (Unterschrift kann nicht durch Dritte erzeugt werden)
  - Transienz (Unterschrift ist nicht wiederverwendbar)
  - Unveränderbarkeit (Unterschrift und Dokument bilden Einheit)
  - Dauerhaftigkeit (Unterschrift kann nicht zurückgezogen werden)
- Lösungen:
  - *XML Signatures* (W3C Recommendation seit 2002-02-12)
  - *SOAP Security Extensions: Digital Signature* (W3C Note)

XML Digital Signatures -- Technik

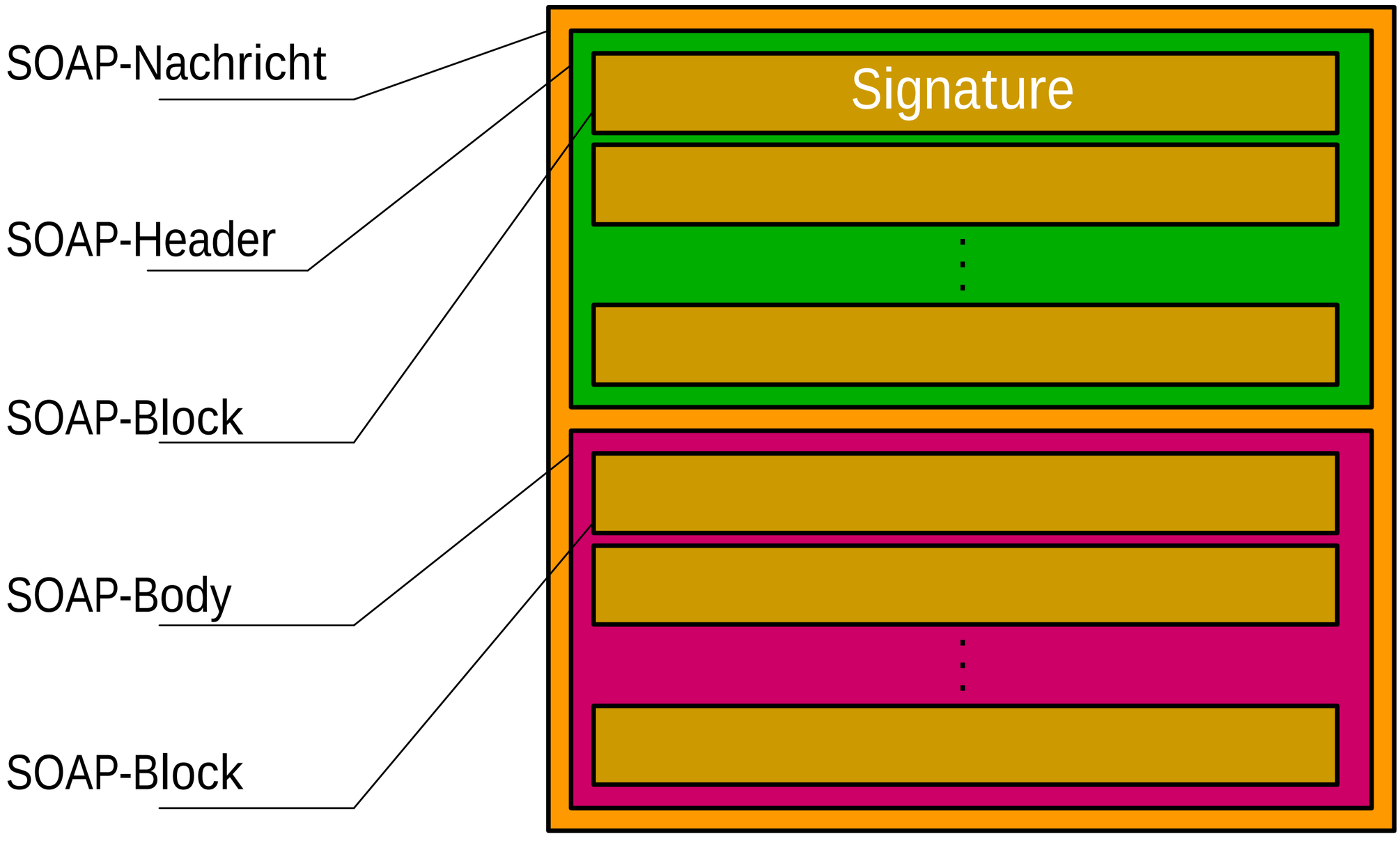
- Ziel: Aufdeckung potentieller Datenverfälschung
  - Sender:



XML Digital Signatures – Unterschriften Prüfvorgang



XML Digital Signatures – Einbettung in eine SOAP-Nachricht



XML Digital Signatures -- Signature Header

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
          Algorithm="http://www.w3.org/TR/2000/WD-xml-c14n-20000119">
        </ds:CanonicalizationMethod>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
         <ds:Reference URI="#Body">
          <ds:Transforms>
            <ds:Transform
               Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-20001026"/>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>zA9itX3gfzCeYIYyalslJezGwAg=</ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>
    MCpXybqBI2Aa3Yx4IovjN3GgchaMDB3lIF1M6a9+Ngi59MqDwK3LRQ==
      </ds:SignatureValue>
</ds:Signature>
```

XML Digital Signatures

SOAP-Nachricht

SOAP-Header

SOAP-Block

SOAP-Body

SOAP-Block

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <SOAP-SEC:Signature
      xmlns:SOAP-SEC="http://schemas.xmlsoap.org/soap/security/2000-12"
      SOAP-ENV:actor="http://example.com/soapEndpoint"
      SOAP-ENV:mustUnderstand="1">
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:SignedInfo>
        <ds:CanonicalizationMethod
        Algorithm="http://www.w3.org/TR/2000/WD-xml-c14n-20000119">
      </ds:CanonicalizationMethod>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1" />
      <ds:Reference URI="#Body">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-20001026" />
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <ds:DigestValue>zA9itX3gfzCeYIYyalslJezGwAg=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>MCpXybqBI2Aa3Yx4IovjN3GgchaMDB3lIF1M6a9+Ngi59MqDwK3LRQ==</ds:SignatureValue>
  </ds:Signature>
    </SOAP-SEC:Signature>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body
    xmlns:SOAP-SEC="http://schemas.xmlsoap.org/soap/security/2000-12"
    SOAP-SEC:id="Body">
      <ns1:add xmlns:ns1="http://www.daimlerchrysler.com/">
        <op1 xsi:type="xsd:int">2</op1>
        <op2 xsi:type="xsd:int">4</op2>
      </ns1:add>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

XML Digital Signatures -- Zusammenfassung

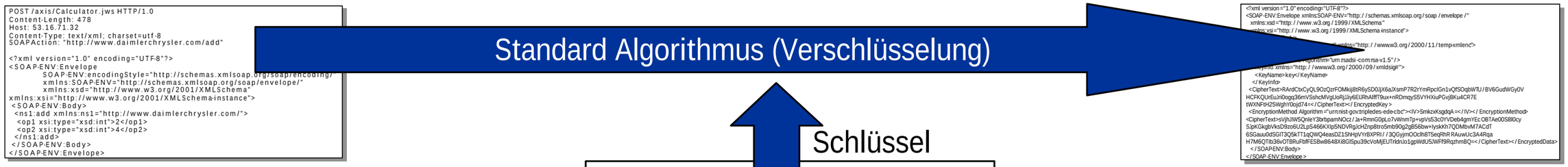
- Ziel: Aufdeckung potentieller Datenverfälschung
- Keine Veränderung des Dateninhaltes (SOAP Body)
- Definition eines Nachrichten *Digests* als Grundlage des Unterschriftenvorganges
- Durch bestehende XML-Lösungen auf XML-Ebene (leicht) lösbar
- Anwendung:
  - Sicherung der Urheberschaft bei nicht-vertraulichen Inhalten (diverse B2B-Anwendungen)
  - Sicherstellung der Unveränderbarkeit (RPCs, Business Transaktionen)

XML Encryption

- Ziel: Schutz der Vertraulichkeit
- Sender verschlüsselt Daten,
  Empfänger entschlüsselt (mit geeignetem Schlüssel)
- Für strengste Anforderungen sinnvollerweise in Applikation auf SOAP-Endpunkt (Aufrufer und Diensterbringer) ausgeführt
- Nicht berücksichtigt:
  - Schlüsselverteilung und –übergabe
  - Verschlüsselungsalgorithmus
- Lösungen:
  - *XML Encryption Syntax and Processing* (W3C Candidate Recommendation seit 2002-03-04)

XML Encryption

- Ziel: Schutz der Vertraulichkeit
  - Sender



XML Dokument

verschlüsselter Datenstrom

- Empfänger

verschlüsselter Datenstrom

XML Dokument

XML Encryption – Fragestellungen

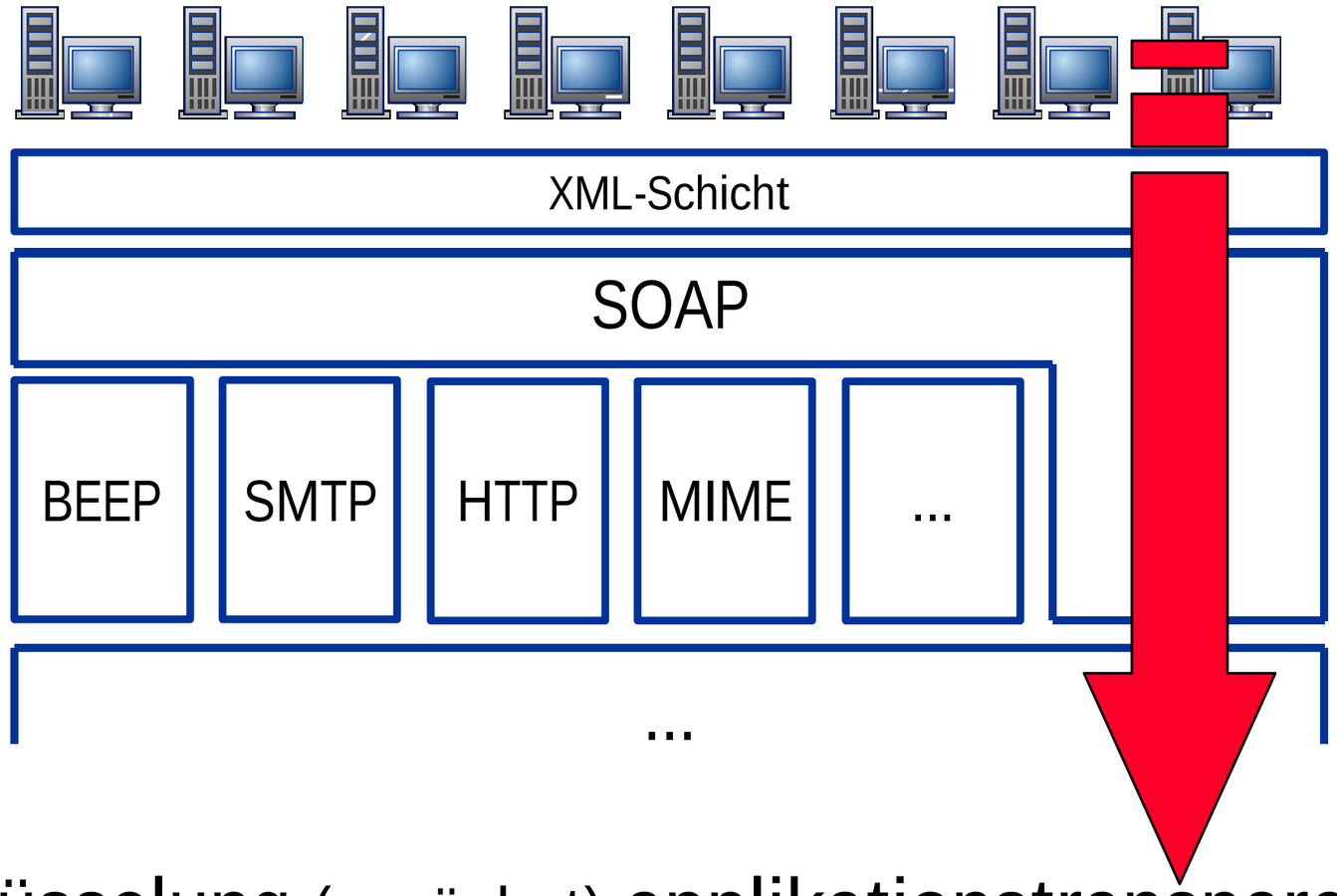
- Auswahl des (geeignetsten) Verschlüsselungsalgorithmus
  (Interoperabilität, Exportbeschränkungen, Schlüssellängen, ...)

- Verwaltung der benötigten Schlüssel
  (z.B. *Public Key Infrastruktur*)

- Einbindung der Verschlüsselungsroutinen in Applikationen,
  oder Bereitstellung eines entsprechenden Dienstes

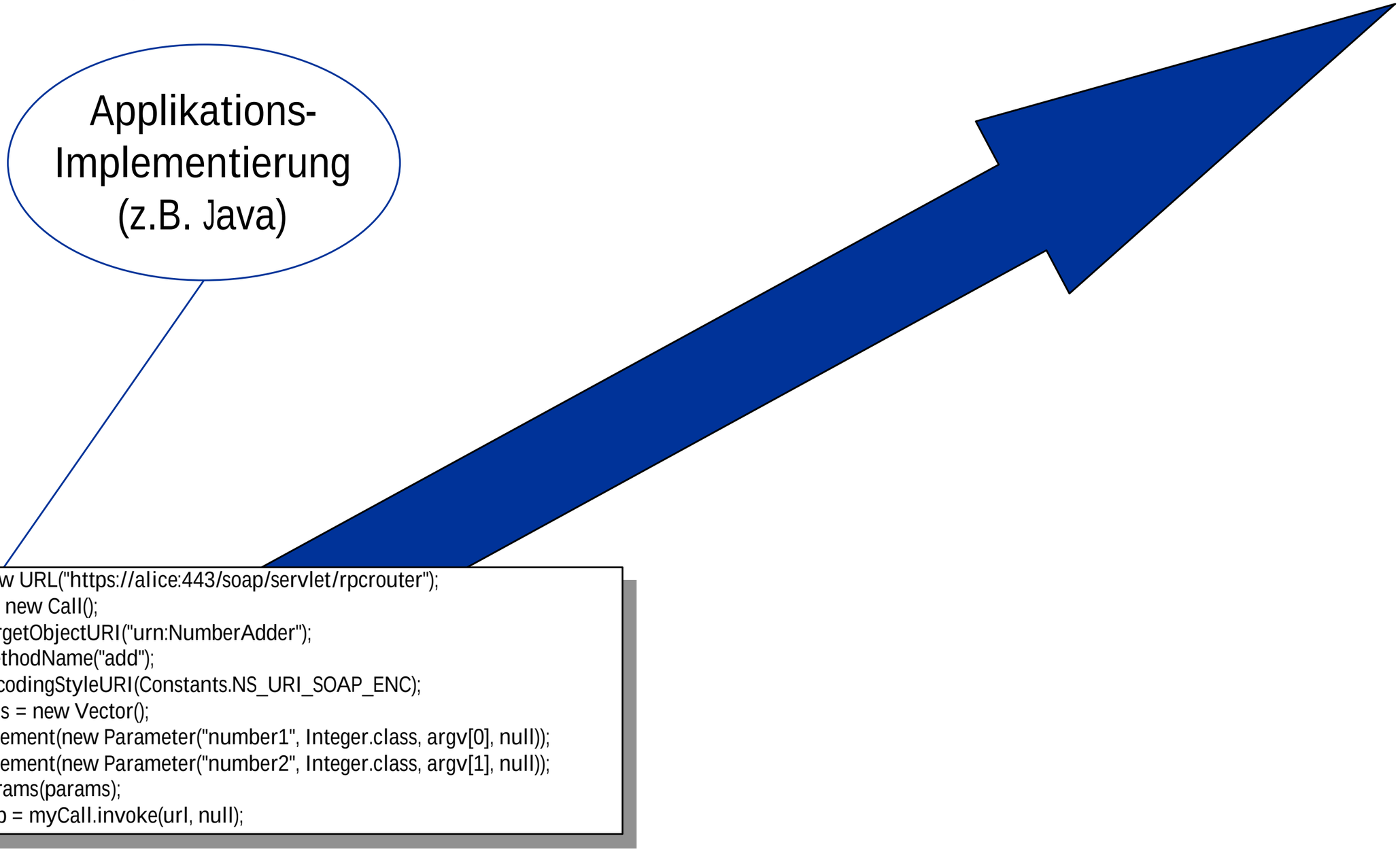
- Festlegung der Verschlüsselungsgranularität
  (gesamter Aufruf, einzelne XML-Elemente, Elementinhalte, ...)

XML Encryption -- Ablauf



- Verschlüsselung (zunächst) applikationstransparent
- Erzeugung des SOAP-Aufruf unverändert
- Verschlüsselung des Aufrufs vor Versendung über Leitung

XML Encryption -- Ablauf



XML Encryption -- Ablauf

SOAP-Aufruf
(Body unverändert)

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:add xmlns:ns1="urn:NumberAdder"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <number1 xsi:type="xsd:int">1</number1>
      <number2 xsi:type="xsd:int">2</number2>
    </ns1:add>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
URL url = new URL("https://alice:443/soap/s
Call myCall = new Call();
myCall.setTargetObjectURI("urn:NumberAdde
myCall.setMethodName("add");
myCall.setEncodingStyleURI(Constants.NS_UR
Vector params = new Vector();
params.addElement(new Parameter("number1", Integer.class, argv[0], null));
params.addElement(new Parameter("number2", Integer.class, argv[1], null));
myCall.setParams(params);
Response resp = myCall.invoke(url, null);
```

XML Encryption -- Ablauf

Verschlüsselter Inhalt (d.h. Body) des SOAP-Aufrufs

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">
  <SOAP-ENV:Body>
    <EncryptedData Type="Element" xmlns="http://www.w3.org/2000/11/temp-xmlenc">
 <EncryptedKey>
  <EncryptionMethod Algorithm="urn:rsadsi-com:rsa-v1.5" />
  <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
   <KeyName>key</KeyName>
  </KeyInfo>

<CipherText>RArdCtxCyQL9OzQzrFOMkij8tR6ySD0JjX6aJXsmP7R2rYmRpcIGn1vQfSOqbWTU/BV6G
udWGy0VHCFKQUrEuJri0ogq36mVSshcMVgUoRjJJiy6EIJRhAIffT9ux+nRDmqyS5VYHXiuPGvjBKu4CR
7EtWXNFtH25WghY0ojd74=</CipherText></EncryptedKey>
 <EncryptionMethod
  Algorithm="urn:nist-gov:tripledes-ede-cbc"><IV>SmkzxKsgdqA=</IV></EncryptionMethod>
<CipherText>sVjhJIW5QnIeY3brbpamNOcz/Ja+RmnG0pLo7vWnmTp+vpVs53c0YVDeb4gmYEcOBTA
e00S8l0cySJpKGkgbVksD9zo6U2LpS466KXIp5NDVRgJcHZnp8tro5mb90g2gB56bw+IyskKh7QDM
bvM7ACdT6SGauu0dSGIT3Q5kTT1qQWQ4easDZ1ShHpVYrBXPRI//3QGyjrnOOclh8T5eqRhRRAuw
Uc3A4RqaH7M6QTIb36vOTBRuFbfFESBw8648Xi8GlSpu39cVoMjEUTrldnJo1gpWdU5JWFf9RqzhmB
Q=</CipherText>
  </EncryptedData>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<?xml version='1.0' enc
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="ht
  xmlns:xsi="http://ww
  xmlns:xsd="http://w
  <SOAP-ENV:Body>
    <ns1:add xmlns:ns1
      SOAP-ENV:encodi
      <number1 xsi:type
      <number2 xsi:type=
    </ns1:add>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
 URL url = new URL("https://alice:443/soap/s
Call myCall = new Call();
myCall.setTargetObjectURI("urn:NumberAdde
myCall.setMethodName("add");
myCall.setEncodingStyleURI(Constants.NS_UR
Vector params = new Vector();
params.addElement(new Parameter("number1", Integer.class, argv[0], null));
params.addElement(new Parameter("number2", Integer.class, argv[1], null));
myCall.setParams(params);
Response resp = myCall.invoke(url, null);
```

XML Encryption

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <SOAP-ENV:Body>
  <ns1:add xmlns:ns1="http://www.daimlerchrysler.com/">
   <op1 xsi:type="xsd:int">2</op1>
   <op2 xsi:type="xsd:int">4</op2>
  </ns1:add>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

XML Encryption

```
<?xml version="1.0" encoding= "UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <EncryptedData Type="Element"
     xmlns="http://www.w3.org/2000/11/temp-xmlenc">
    <EncryptedKey>
     <EncryptionMethod Algorithm="urn:rsadsi-com:rsa-v1.5"/>
     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
      <KeyName>key</KeyName>
     </KeyInfo>


  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

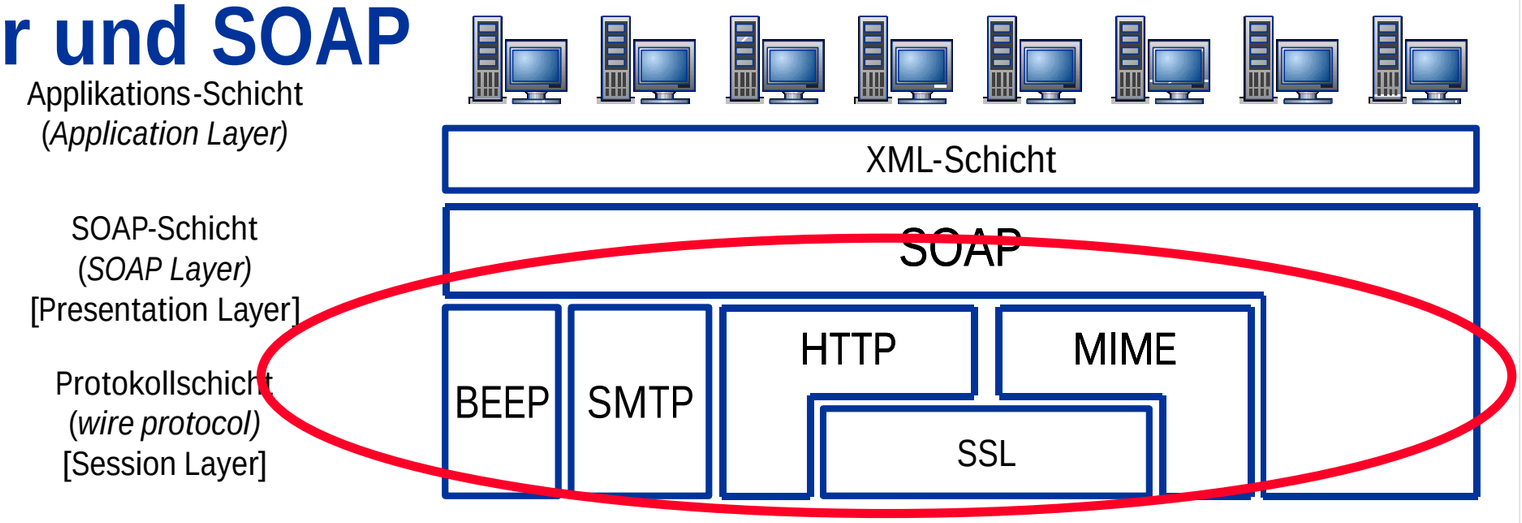
XML Encryption

```
<?xml version= "1.0" encoding= "UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <EncryptedData Type="Element"
    xmlns="http://www.w3.org/2000/11/temp-xmlenc">
      <EncryptedKey>
        <EncryptionMethod Algorithm="urn:rsadsi-com:rsa-v1.5"/>
          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <KeyName>key</KeyName>
          </KeyInfo>
        <CipherText>RArdCtxCyQL9OzQzrFOMkij8tR6ySD0JjX6aJXsmP7R2rYmRpcIGn1vQfSOqbWTU/BV6GudWGy0V
        HCFKQUrEuJri0ogq36mVSshcMVgUoRjJJiy6EIJRhAIffT9ux+nRDmqyS5VYHXiuPGvjBKu4CR7E
        tWXNFtH25WghY0ojd74=</CipherText>
      </EncryptedKey>
      <EncryptionMethod
        Algorithm="urn:nist-gov:tripledes-ede-cbc">
        <IV>SmkzxKsgdqA=</IV>
      </EncryptionMethod>
      <CipherText>sVjhJlW5QnIeY3brbpamNOcz/Ja+RmnG0pLo7vWnmTp+vpVs53c0YVDeb4gmYEcOBTAe00S8l0cy
      SJpKGkgbVksD9zo6U2LpS466KXIp5NDVRgJcHZnp8tro5mb90g2gB56bw+IyskKh7QDMbvM7ACdT
      6SGauu0dSGIT3Q5kTT1qQWQ4easDZ1ShHpVYrBXPRI//3QGyjrnOOclh8T5eqRhRRAuwUc3A4Rqa
      H7M6QTIb36vOTBRuFbfFESBw8648Xi8GISpu39cVoMjEUTrIdnJo1gpWdU5JWFf9RqzhmBQ=</CipherText>
      </EncryptedData>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

XML Encryption -- Zusammenfassung

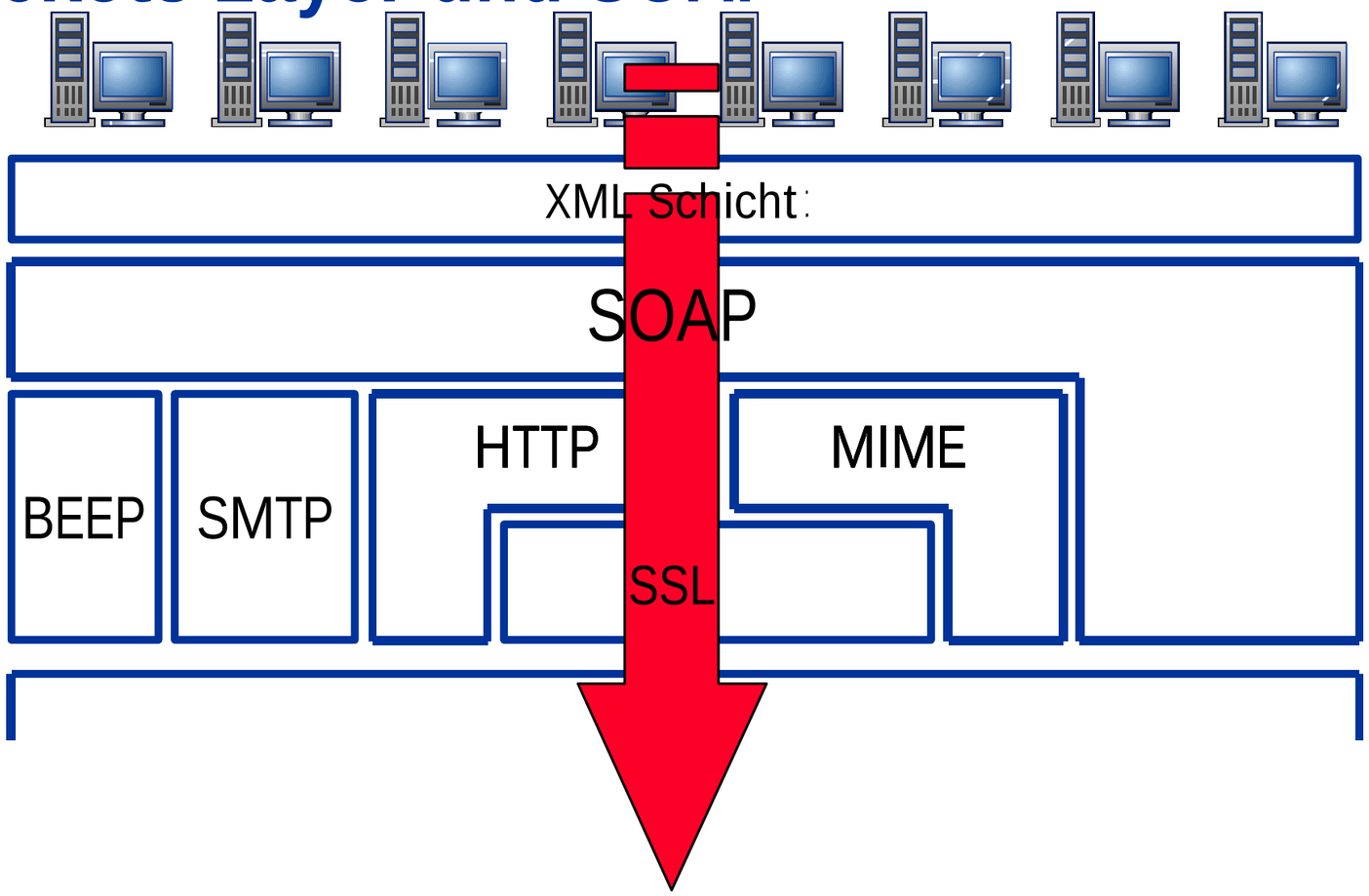
- Ziel: Schutz der Vertraulichkeit
- Frei wählbare Granularität der Verschlüsselung
  (Element, Elementinhalt, vollständiger Teilbaum, ...)
- Frei wählbarer Verschlüsselungsalgorithmus
- Struktur-zerstörende Transformation
  (Verschlüsseltes Dokument ist jedoch noch *schema-valid* SOAP)
- XML-externe Schlüsselverwaltung notwendig
- Eingriff in Applikationscode
  (oder ggf. entsprechender Service)

Secure Sockets Layer und SOAP



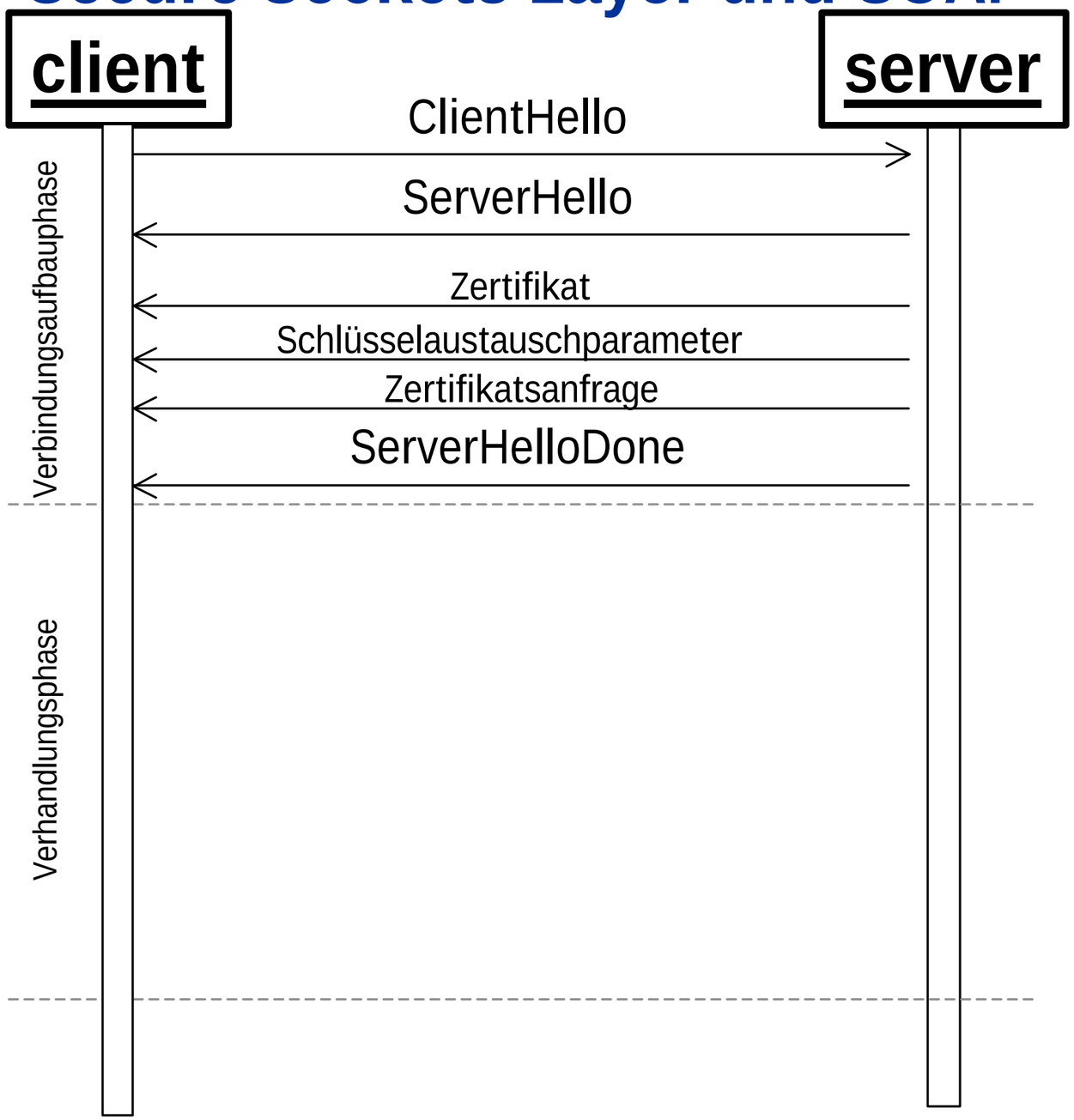
- Ziel: Transparente Sicherung unterhalb der Darstellungsschicht
- Bekannteste Anwendung: Mit HTTP zu HTTPS kombiniert
- (ursprünglich) zur Sicherung von HTTP-Verbindungen konzipiert
- Durch Web-Server implementiert (z.B. mod_ssl) und gängige Browser unterstützt (z.B. NC, IE, Opera, Mozilla, Lynx, ...)
- Einsatz für XML vollkommen, und für SOAP praktisch transparent

Secure Sockets Layer und SOAP



- SOAP-Aufruf unverändert; SOAP-Endpoint Port 443
- Durch Zertifikatsaustausch auf längerfristige Kommunikation ausgelegt
- Inhalte der Transportschicht werden verschlüsselt übertragen

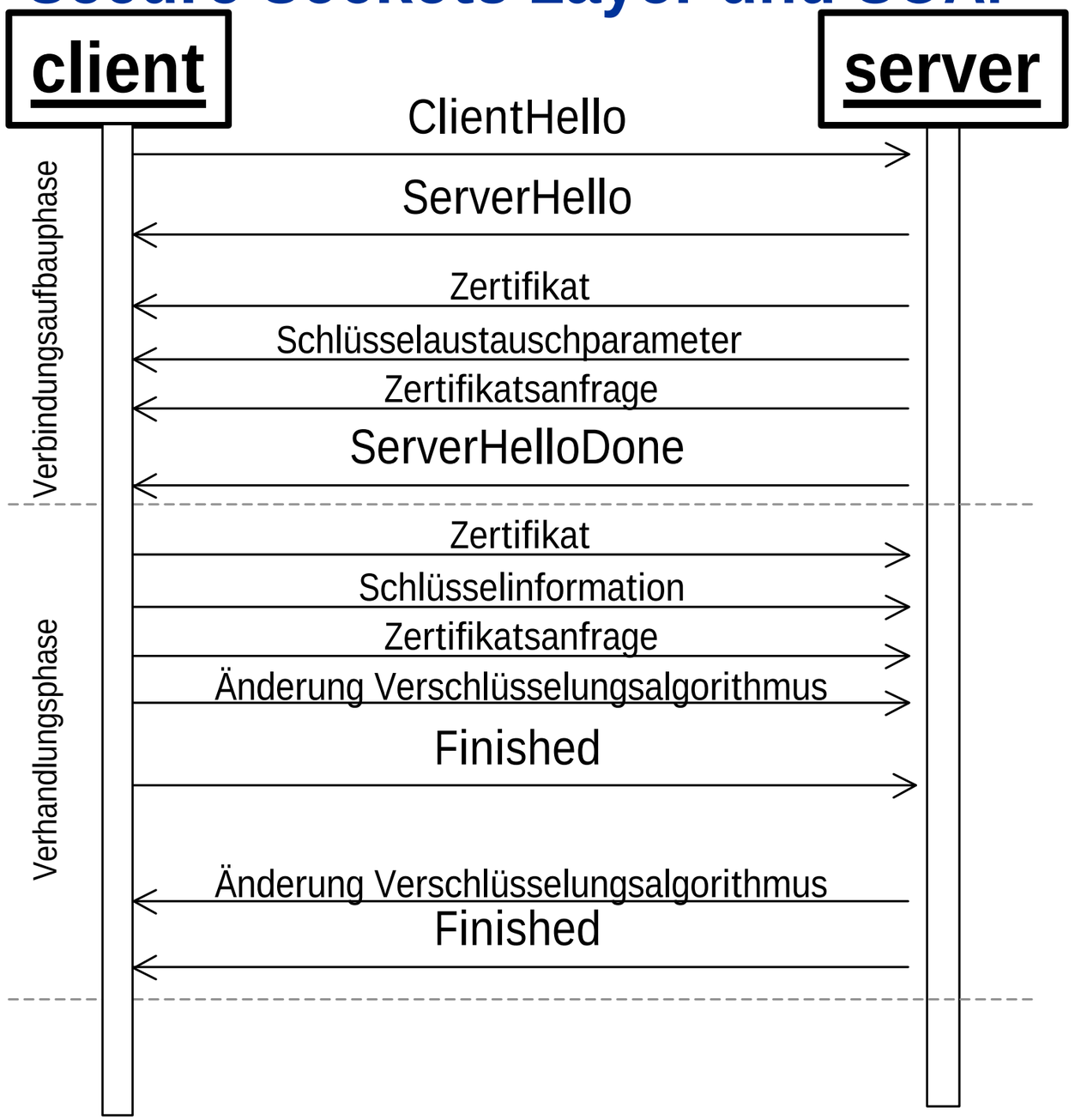
Secure Sockets Layer und SOAP – *SSL handshake*



ClientHello umfaßt:
- Geordnete Liste unterstützter Krypto-Algorithmen
- Geordnete Liste unterstützter Kompressionsmethoden

ServerHello umfaßt:
- Gewählter Krypto-Algorithmus (aus Client-Liste)
- Gewählte Kompressionsmethode (aus Client-Liste)
- Server-Zertifikat (optional)
- Parameter für den Schlüsselaustausch (optional)
- Zertifikatsanfrage (nach Client-Zertifikat (optional))

Secure Sockets Layer und SOAP – *SSL handshake*



ClientHello umfaßt:
- Geordnete Liste unterstützter Krypto-Algorithmen
- Geordnete Liste unterstützter Kompressionsmethoden

ServerHello umfaßt:
- Gewählter Krypto-Algorithmus (aus Client-Liste)
- Gewählte Kompressionsmethode (aus Client-Liste)
- Server-Zertifikat (optional)
- Parameter für den Schlüsselaustausch (optional)
- Zertifikatsanfrage (nach Client-Zertifikat (optional))

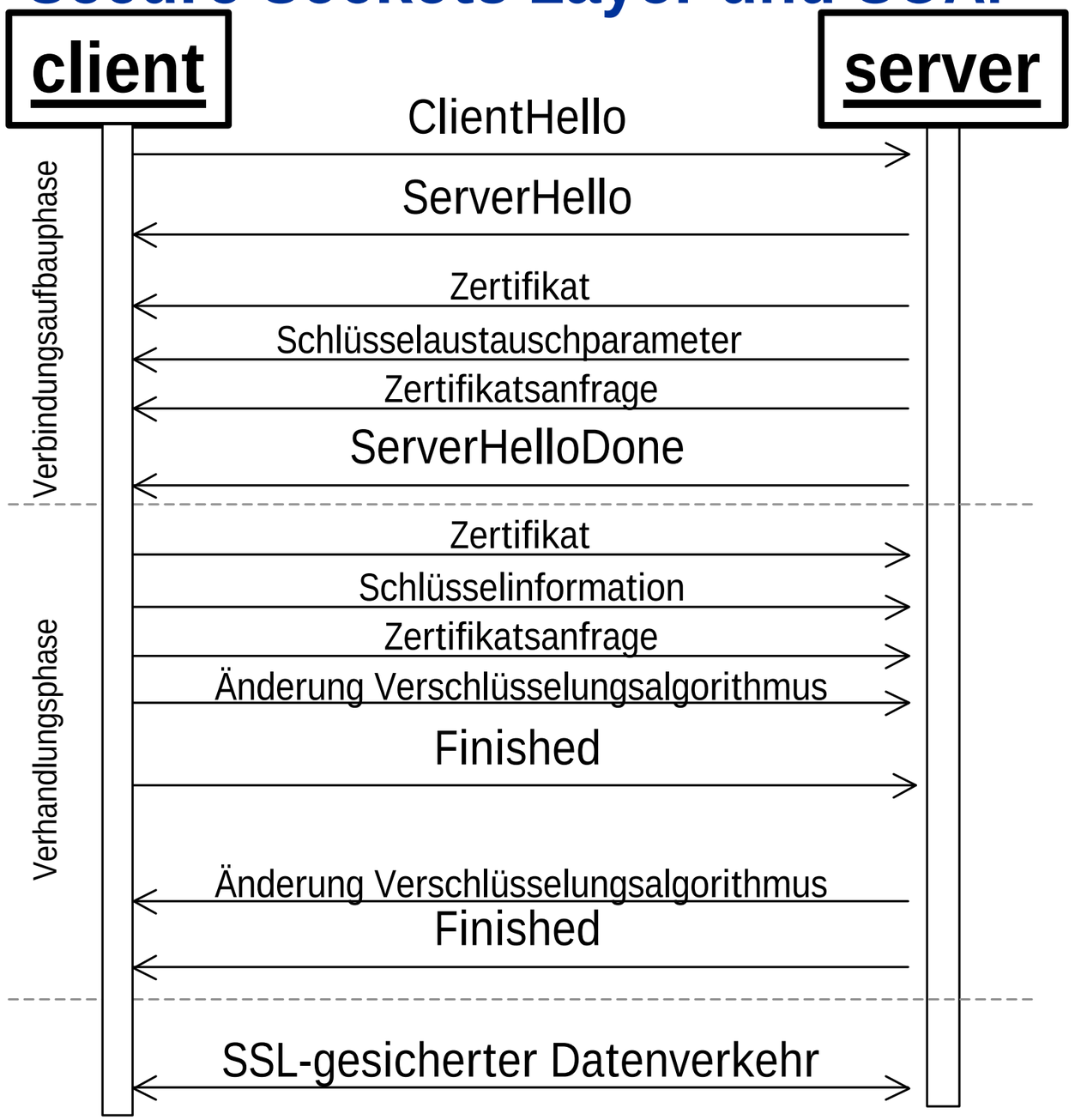
Abstimmung bezüglich:
- Protokollversion
- Verschlüsselungsalgorithmus
- Gegenseitige Authentifizierung (optional) mit public key Techniken

Finish umfaßt:
- Client-Zertifikat (optional)
- Client-Schlüssel (optional)
- Ergebnis der Server-Zertifikats-Prüfung (optional)

- Evtl. Abänderung des vorgeschlagenen Verschlüsselungsalgorithmus

(kann durch Client und Server gleichermaßen geschehen)

Secure Sockets Layer und SOAP – *SSL handshake*



ClientHello umfaßt:
- Geordnete Liste unterstützter Krypto-Algorithmen
- Geordnete Liste unterstützter Kompressionsmethoden

ServerHello umfaßt:
- Gewählter Krypto-Algorithmus (aus Client-Liste)
- Gewählte Kompressionsmethode (aus Client-Liste)
- Server-Zertifikat (optional)
- Parameter für den Schlüsselaustausch (optional)
- Zertifikatsanfrage (nach Client-Zertifikat (optional))

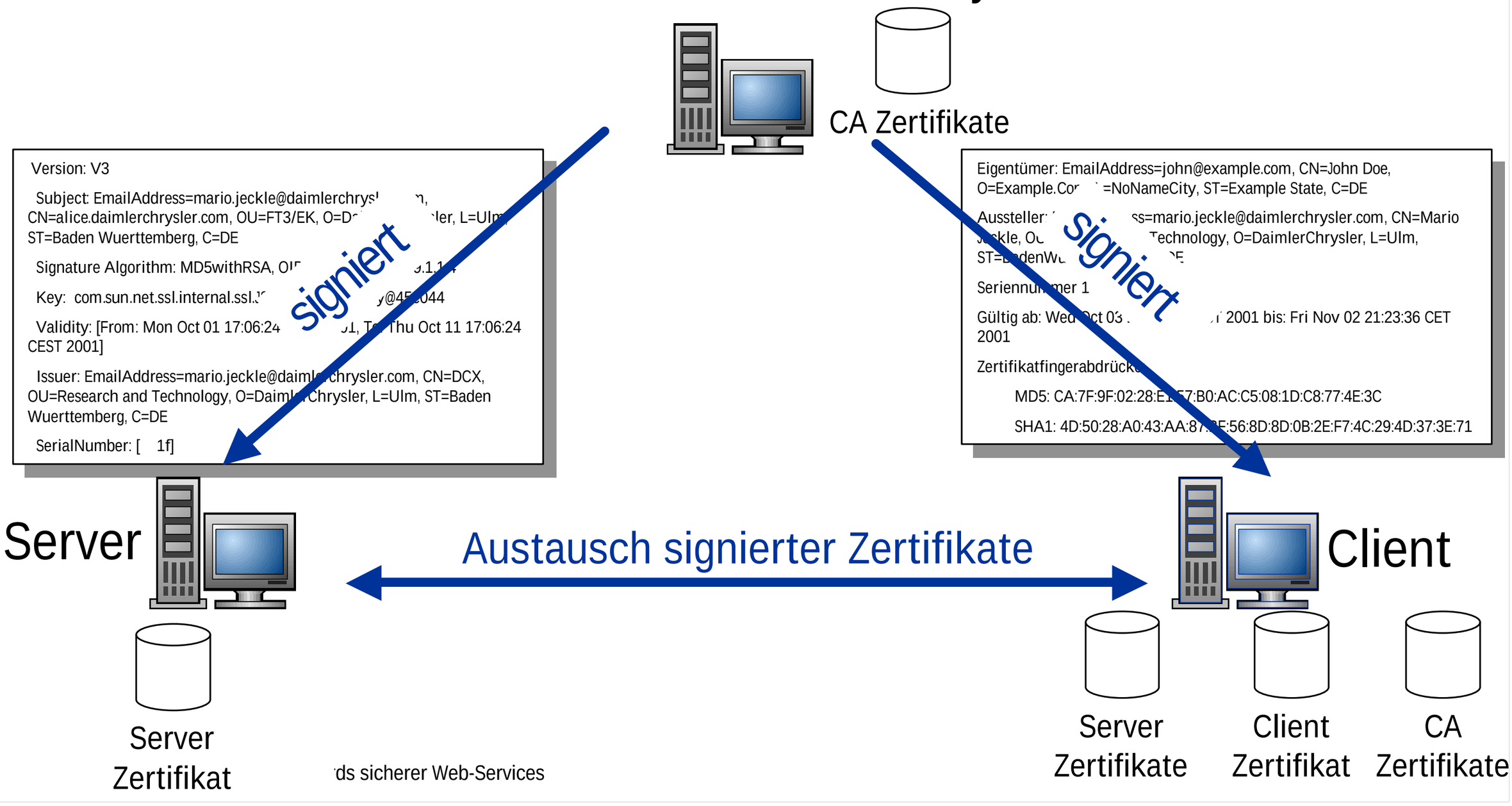
Abstimmung bezüglich:
- Protokollversion
- Verschlüsselungsalgorithmus
- Gegenseitige Authentifizierung (optional) mit public key Techniken

Finish umfaßt:
- Client-Zertifikat (optional)
- Client-Schlüssel (optional)
- Ergebnis der Server-Zertifikats-Prüfung (optional)

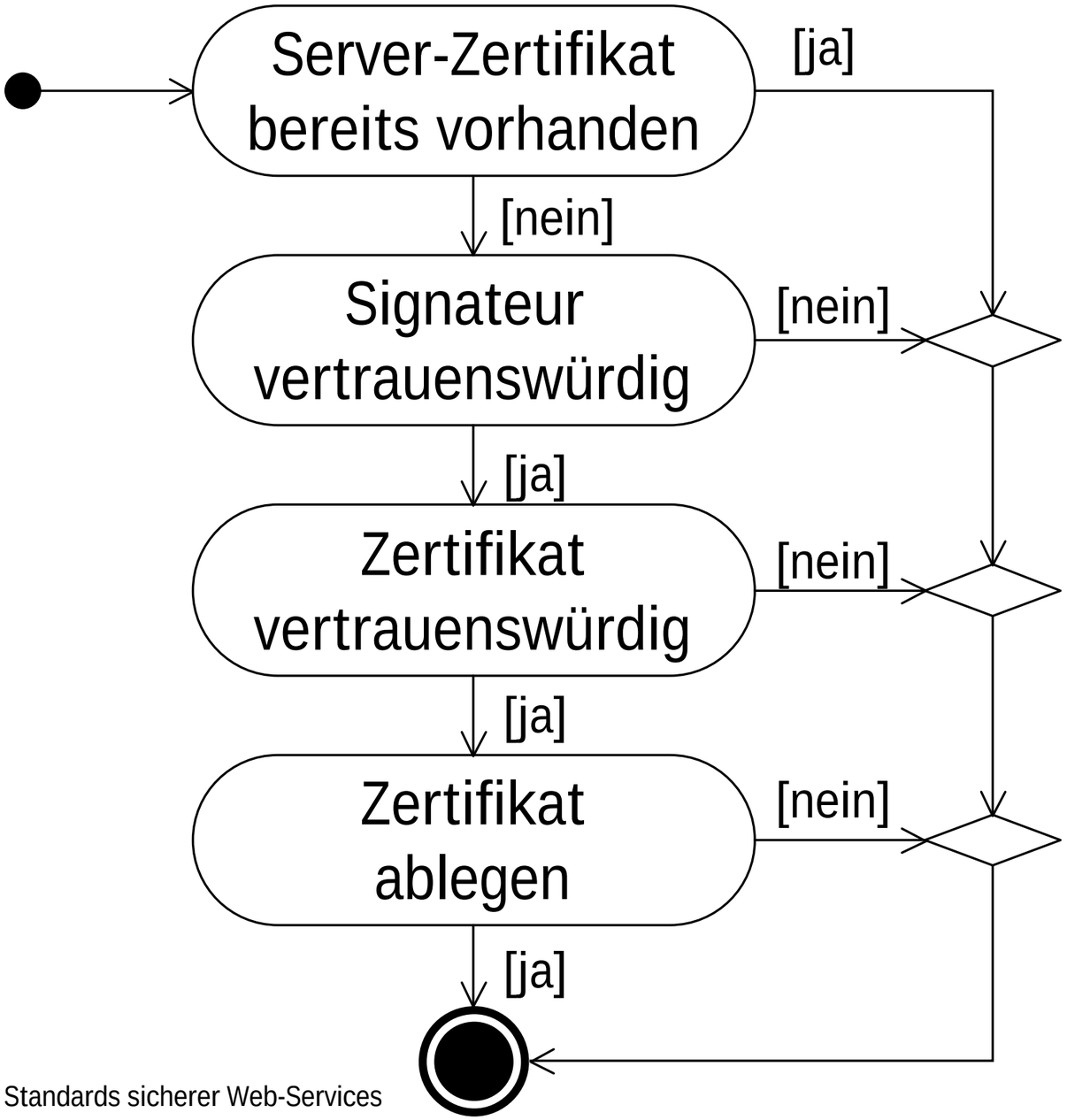
- Evtl. Abänderung des vorgeschlagenen Verschlüsselungsalgorithmus

(kann durch Client und Server gleichermaßen geschehen)

Secure Sockets Layer und SOAP -- Zertifikate



Secure Sockets Layer und SOAP – *SSL handshake*



Secure Sockets Layer und SOAP – Beispiel: handshake

```
KeyManager []km = null;
TrustManager []tma = {new MyX509TrustManager()};
SSLContext sslContext = SSLContext.getInstance("SSL");
sslContext.init(km,tma,new java.security.SecureRandom());
SSLSocketFactory sf = sslContext.getSocketFactory();
Socket sock=new Socket("alice",443);
SSLSocket sslsock=(SSLSocket)sf.createSocket(sock, "alice", 443, true);
sslsock.startHandshake();

javax.securtiy.cert.X509Certificate[] c = sslsock.getSession().getPeerCertificateChain();
ByteArrayOutputStream baos = new ByteArrayOutputStream(1024);
baos.write(c[0].getEncoded(), 0, (c[0].getEncoded()).length );
ByteArrayInputStream bais =
  new ByteArrayInputStream(baos.toByteArray(), 0, (c[0].getEncoded()).length);
CertificateFactory cf = CertificateFactory.getInstance("X.509");
java.security.cert.Certificate cert = cf.generateCertificate(bais);
```

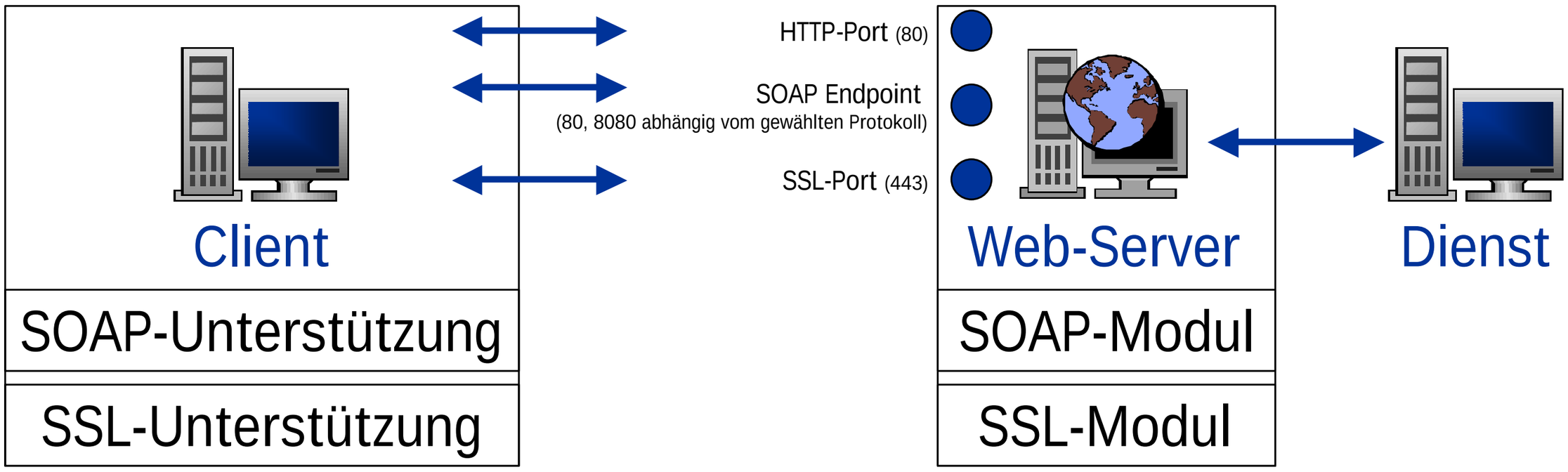
Secure Sockets Layer und SOAP – Beispiel: Dienstaufruf

```
URL url = new URL("https://alice:443/soap/servlet/rpcrouter");
Call myCall = new Call();
myCall.setTargetObjectURI("urn:NumberAdder");
myCall.setMethodName("add");

myCall.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
Vector params = new Vector();

params.addElement(new Parameter("number1", Integer.class, argv[1], null));
params.addElement(new Parameter("number2", Integer.class, argv[1], null));
myCall.setParams(params);

 Response resp = myCall.invoke(url, null);
```

Secure Sockets Layer und SOAP

- Implementierung des SSL handshake muß durch Applikation erfolgen.
  Keine Integration in SOAP-Aufruf möglich!
- Organisatorischer Aufwand durch Beschaffung CA-signierter Server-Zertifikate
- Durchführung des SSL handshake vor jedem SOAP-Aufruf hat Auswirkungen auf Laufzeitverhalten.
  Negativ insbesondere bei einmaliger Kommunikation
- u.U. Strategie zur Ablage der Serverzertifikate sinnvoll oder gar organisatorisch notwendig

Organisatorisch- / Technische Voraussetzungen



- SOAP-Unterstützung (Client- und Server-seitig)
- Freischaltung SSL-Port
- SSL-Unterstützung (Client- und Server-seitig)

Vergleich HTTPS (SSL) und S-HTTP

HTTPS (HTTP über SSL)	S-HTTP (secure HTTP)
Langfristige Kommunikationsbeziehung	Spontankommunikation
Transportprotokoll	Applikationsprotokoll
Eingriff in Firewall-Regeln notwendig	Firewall-Regeln unverändert
Fixierte Sicherheitsstufe	Frei wählbare Sicherheitsstufe
Vergleichsweise leichte Implementier- und Administrierbarkeit	u. U. komplexe Administration
Große Verbreitung und Unterstützung	Kaum verbreitet

Zusammenfassung -- Erfüllung der Anforderungen

- Vertraulichkeit (confidentiality)
  - XML Encryption
  - SSL
- Berechtigung (authorization)
  - XML Digital Signature
  - SSL
- (Daten-)konsistenz (data integrity)
  - XML Digital Signature
  - SSL
- Glaubwürdigkeit des Ursprungs (message origin authentication)
  - XML Digital Signature
  - SSL
- Verbindlichkeit (non-repudiation)
  - XML Digital Signature
  - SSL

Sicheres SOAP in der Praxis

- Im Extranet- / Internetverkehr sollten alle (business-)Nachrichten und RPCs (zumindest) digital signiert werden
- Kombination von digitaler Signatur und SSL kein Widerspruch
- SSL-Verschlüsselung ist zumeist sehr schwach, daher Kombination von SSL und XML Encryption u. U. sinnvoll
- SOAP-Sicherheit sollte im Kontext einer Public Key Infrastruktur (RFC 2459) mitberücksichtigt werden
- Teilweise reichen die vorgestellten Mechanismen nicht aus ... Zusätzlich Einsatz von Sicherheitsmechanismen tieferliegender Protokollebenen möglich (z.B. IPSEC)

Werkzeuge und Applikationen

- Apache SOAP v2.1: http://xml.apache.org
- Servlet Engine: Tomcat: http://xml.apache.org
- SUN JSSE-API: http://java.sun.com
- IBM JSSE-Implementierung: http://www.ibm.com
- XML Digital Signatures: http://www.w3.org/TR/xmldsig-core/
- XML Encryption: http://www.w3.org/TR/xmlenc-core/
- Secure Sockets Layer (IETF Draft): draft-freier-ssl-version3-02
- SOAP Security Extensions: http://www.w3.org/TR/SOAP-dsig/

Dieser Vortrag und weiterführende Information zum Thema:

- http://www.jeckle.de