7363 - Web-basierte Anwendungen

Eine Vertiefungsveranstaltung
mit Schwerpunkt auf XML-Technologien

Web Services Security

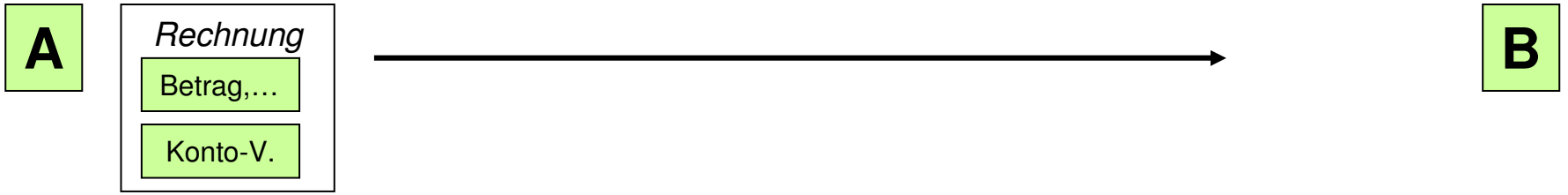
Eine kleine Einführung

Sicherheitsprobleme

Grundsätzliches zu PKI am Beispiel WS

Szenario

- Normalfall
  - A sendet strukturierte Geschäftsdaten an B per WS

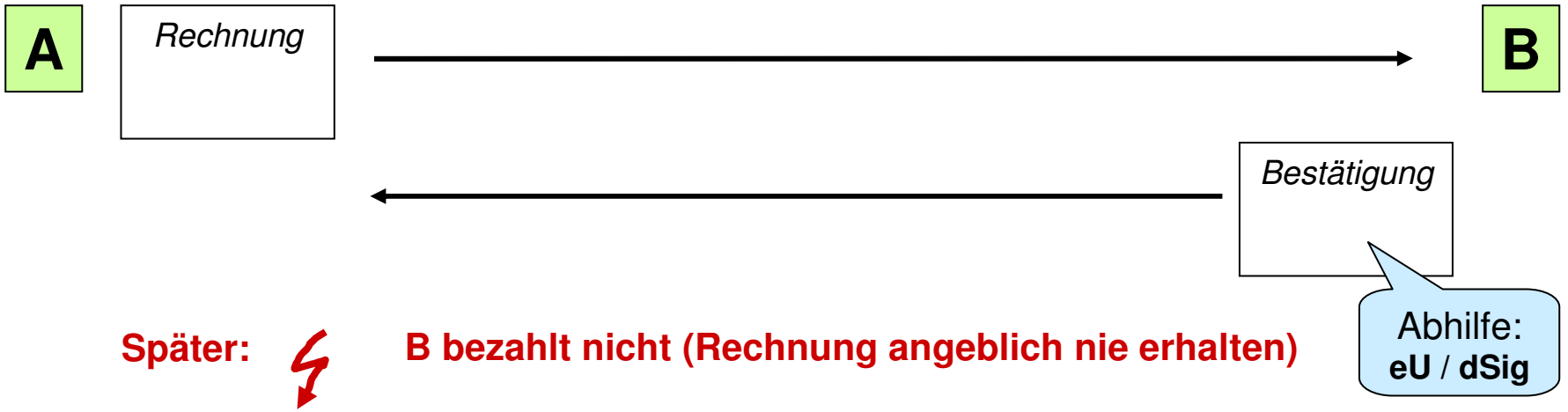


  - Optional: B generiert & antwortet mit Empfangsbestätigung

Szenario

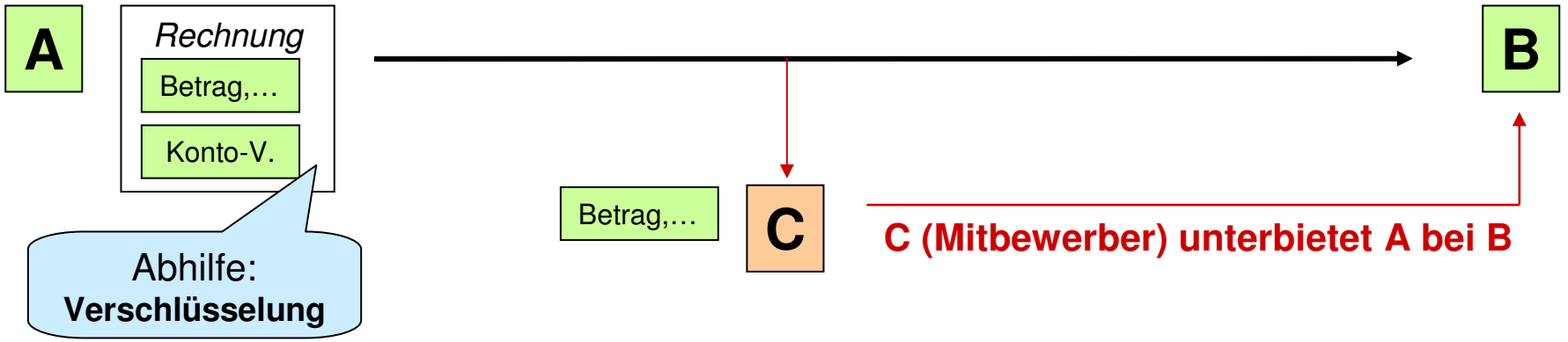
- Störfälle
  - A bestreitet Versendung (*repudiation of origin*)

  - B bestreitet den Versand der Bestätigung (*repudiation of receipt*)

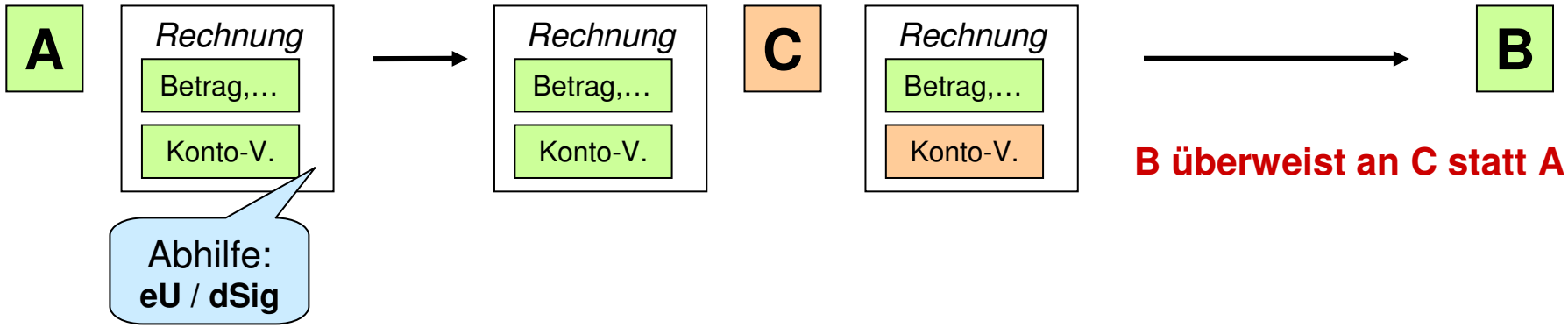


Szenario

- Störfälle
  - C liest mit (Bruch der Vertraulichkeit der Daten)

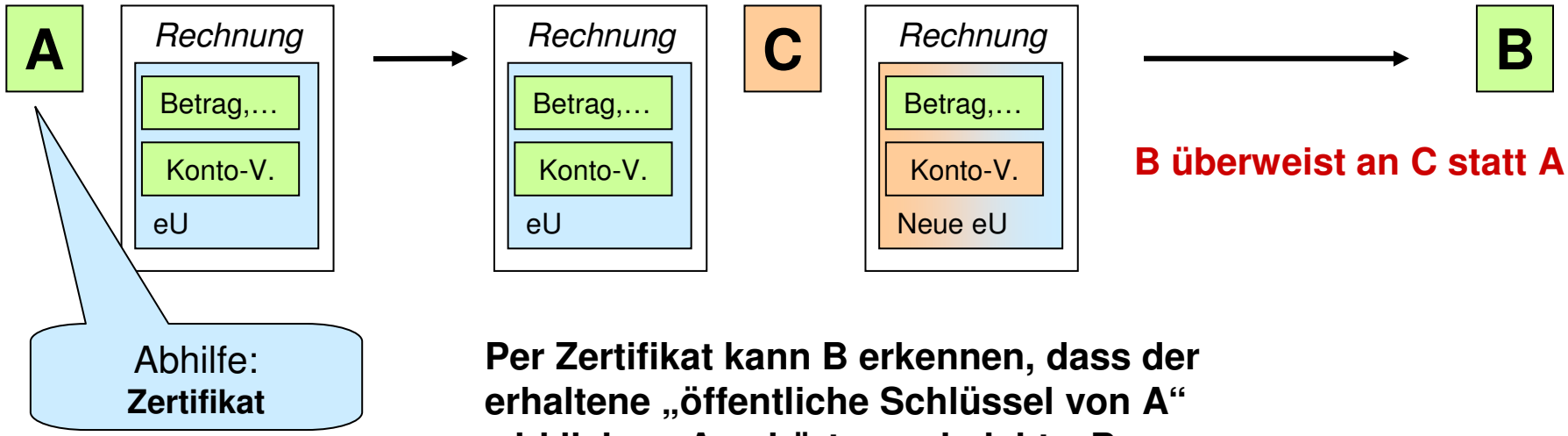


  - C fängt ab & verändert Daten, leitet an B weiter (Bruch der Integrität)



Szenario

- Störfälle
  - C fängt ab & verändert Daten, leitet an B weiter (Bruch d. Integrität), und:
  - C gibt sich B gegenüber als A aus (Bruch der Authentizität)



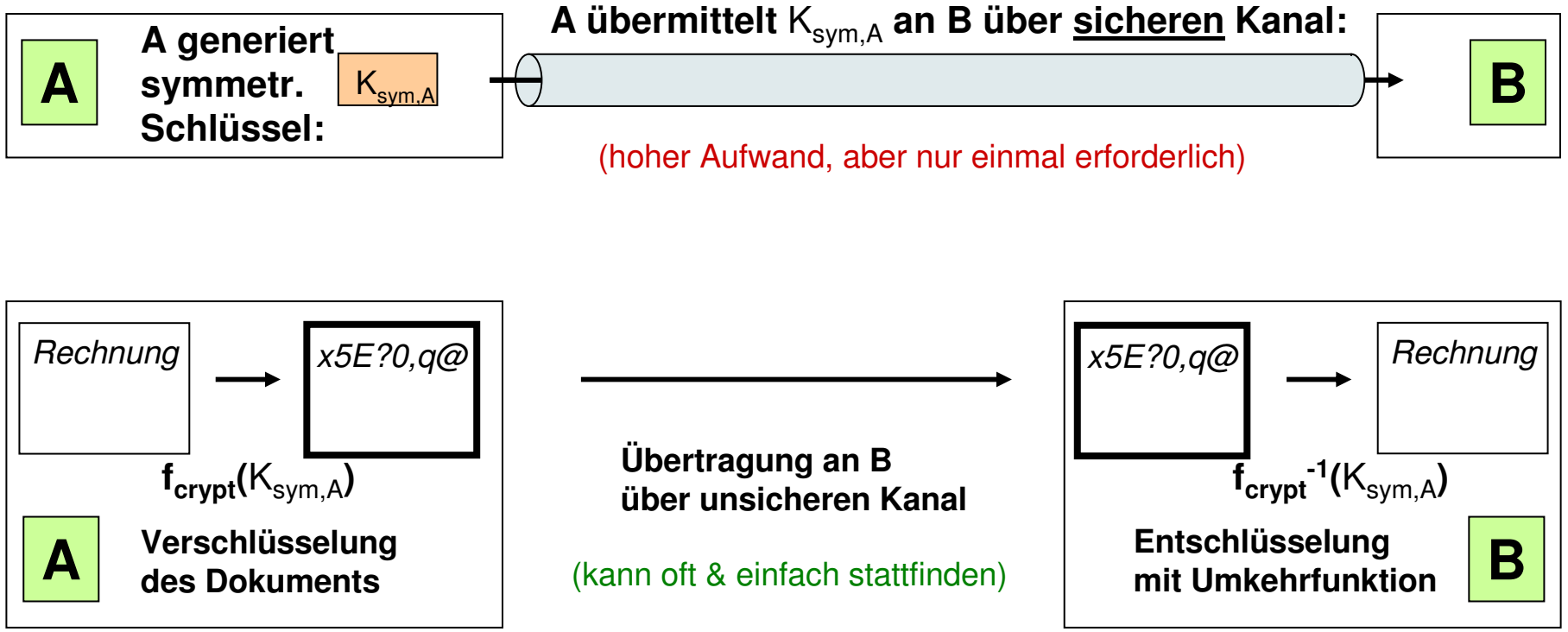
Per Zertifikat kann B erkennen, dass der erhaltene „öffentliche Schlüssel von A“ wirklich zu A gehört – und nicht z.B. von C untergeschoben wurde.

Datenschutz und Kryptografie: Technische Grundlagen

Symmetrische Verschlüsselung
Asymmetrische Verschlüsselung
Hash-Funktionen, digitale Signatur
Zertifikate

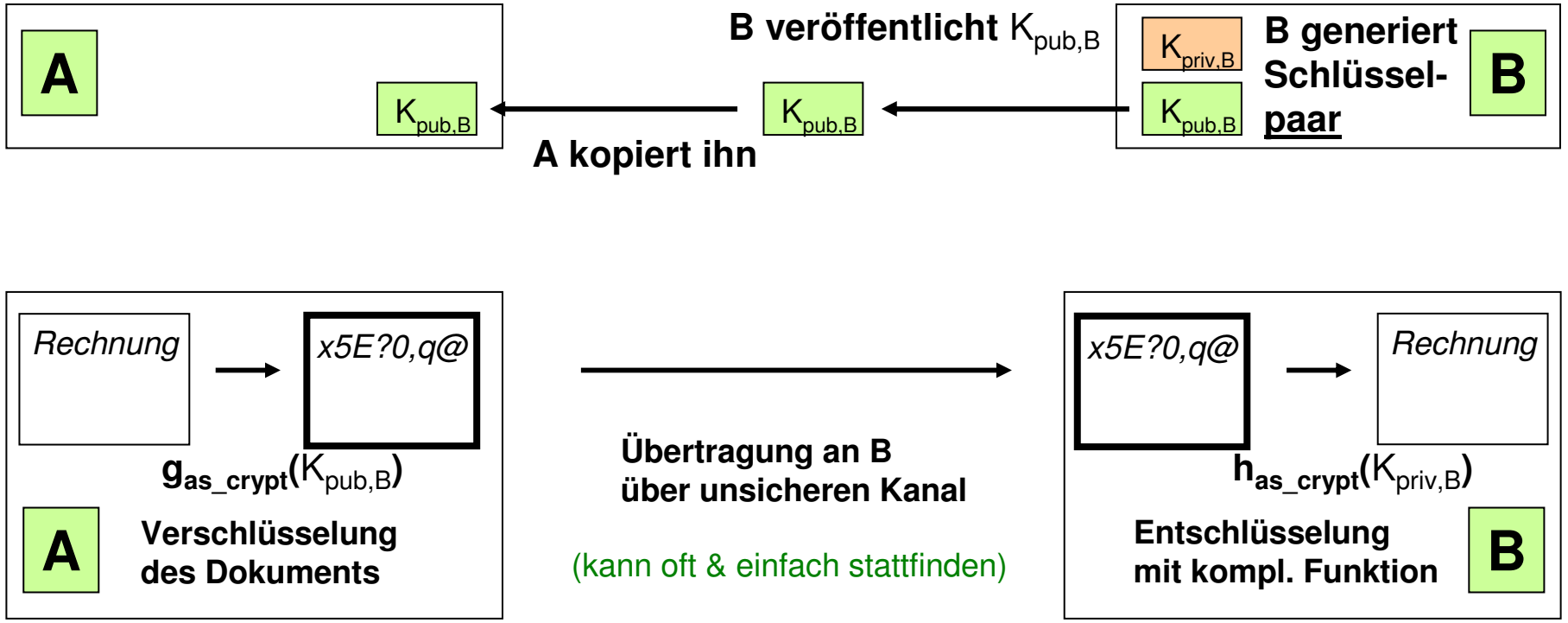
Technische Grundlagen

- Prinzip der symmetrischen Verschlüsselung
  - Situation: A möchte vertrauliche Nachrichten an B senden
  - Vorgehen:



Technische Grundlagen

- Prinzip der asymmetrischen Verschlüsselung
  - Situation: A möchte ohne sicheren Kanal vertrauliche Nachrichten an B senden.
  - Vorgehen:

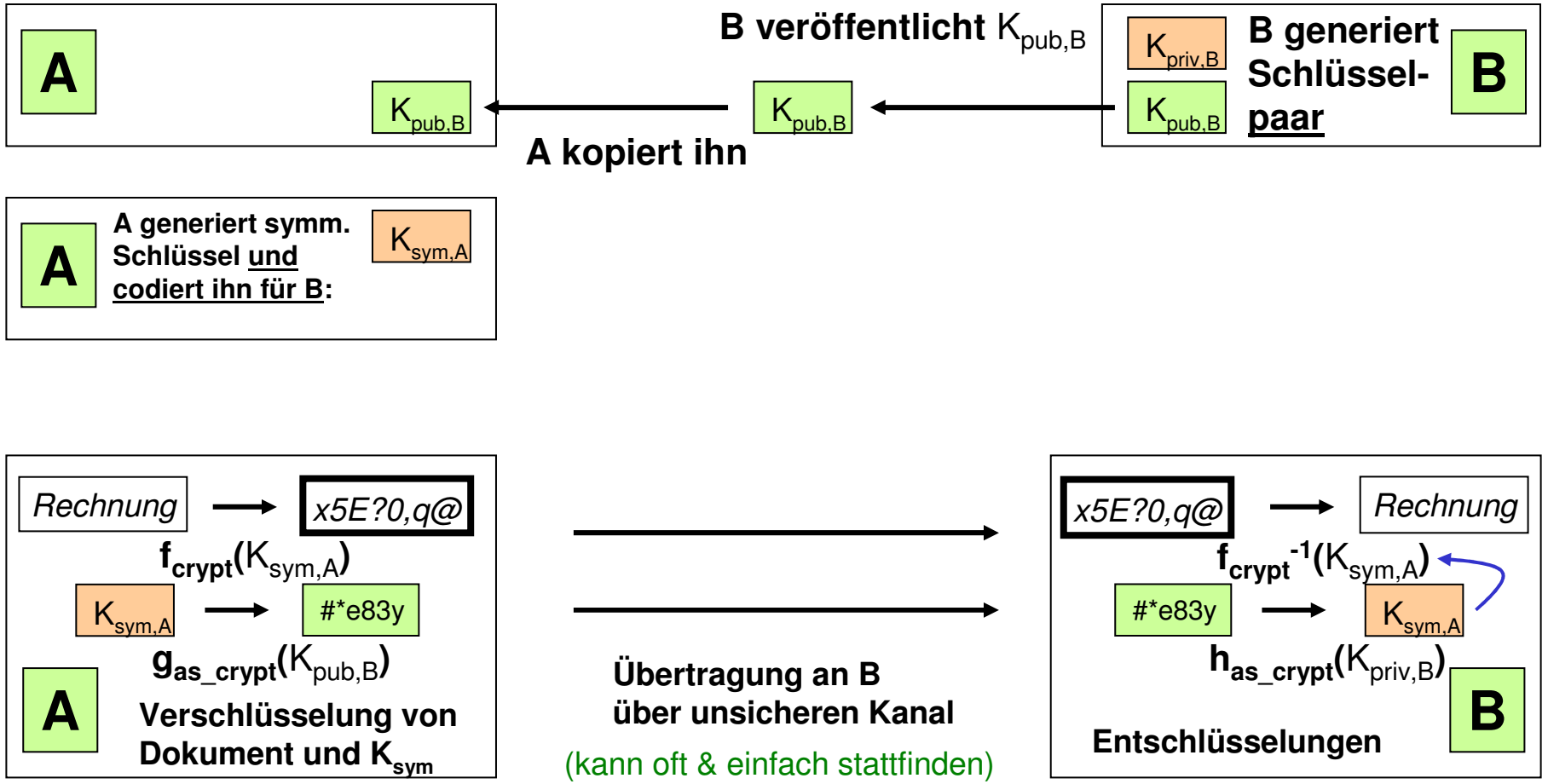


Nachteil:

Funktionen g_{as_crypt} und h_{as_crypt} sind viel rechenaufwändiger als f_{crypt} bzw. f_{crypt}^{-1}

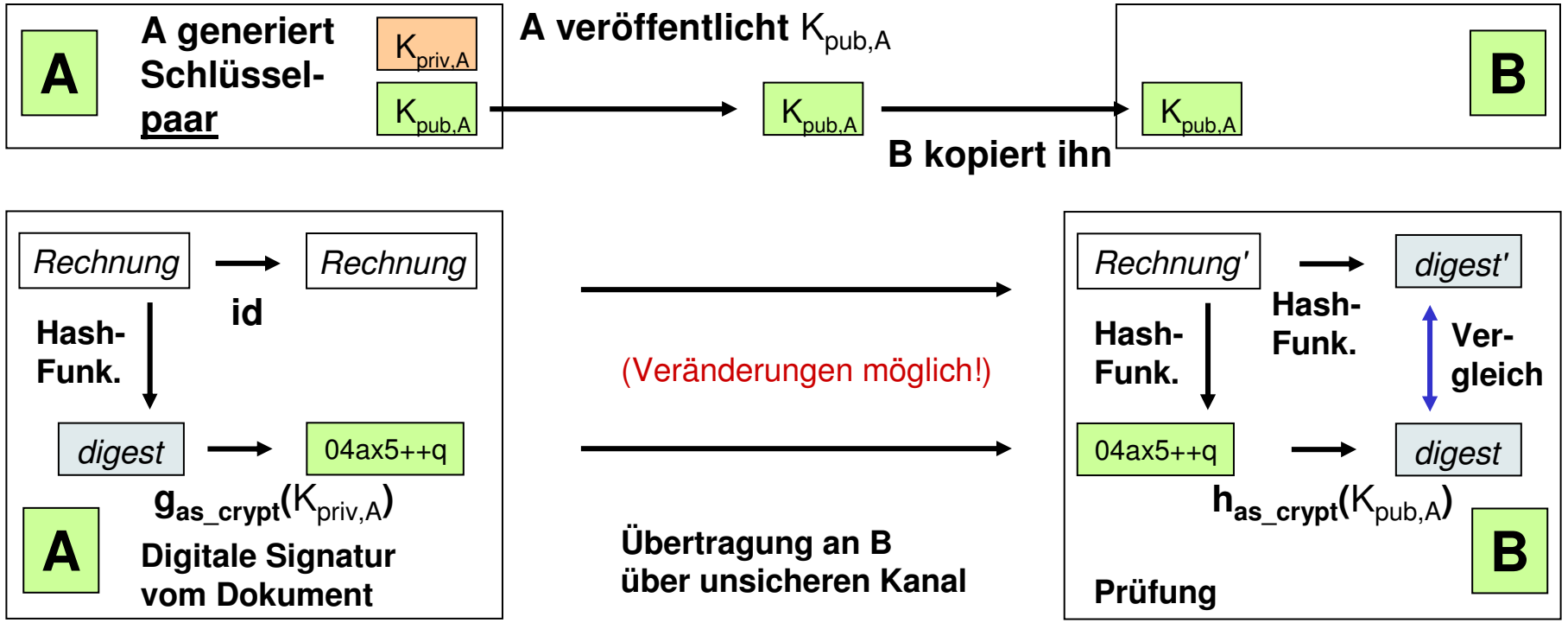
Technische Grundlagen

- Prinzip der effizienten asymmetrischen Verschlüsselung
  - Situation: A möchte ohne sicheren Kanal vertrauliche Nachrichten an B senden und effizient codieren/decodieren. Vorgehen: Nur symm. Schlüssel aufwändig codieren!



Technische Grundlagen

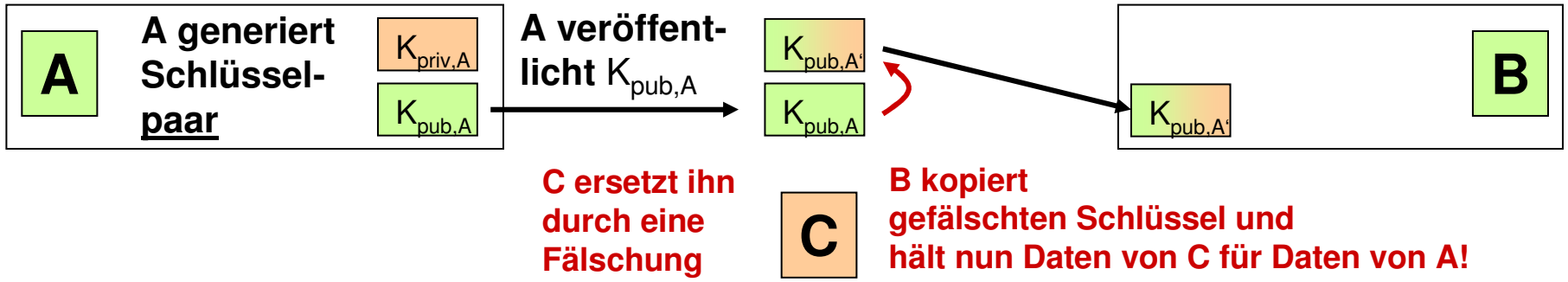
- Hash-Funktionen (*digests*) und digitale Signatur (eU)
  - Situation: A möchte garantieren, dass seine Nachrichten B unverändert erreichen.
  - Vorgehen: *Digest* der Inhalte berechnen und mit eigenen priv. Schlüssel codieren.



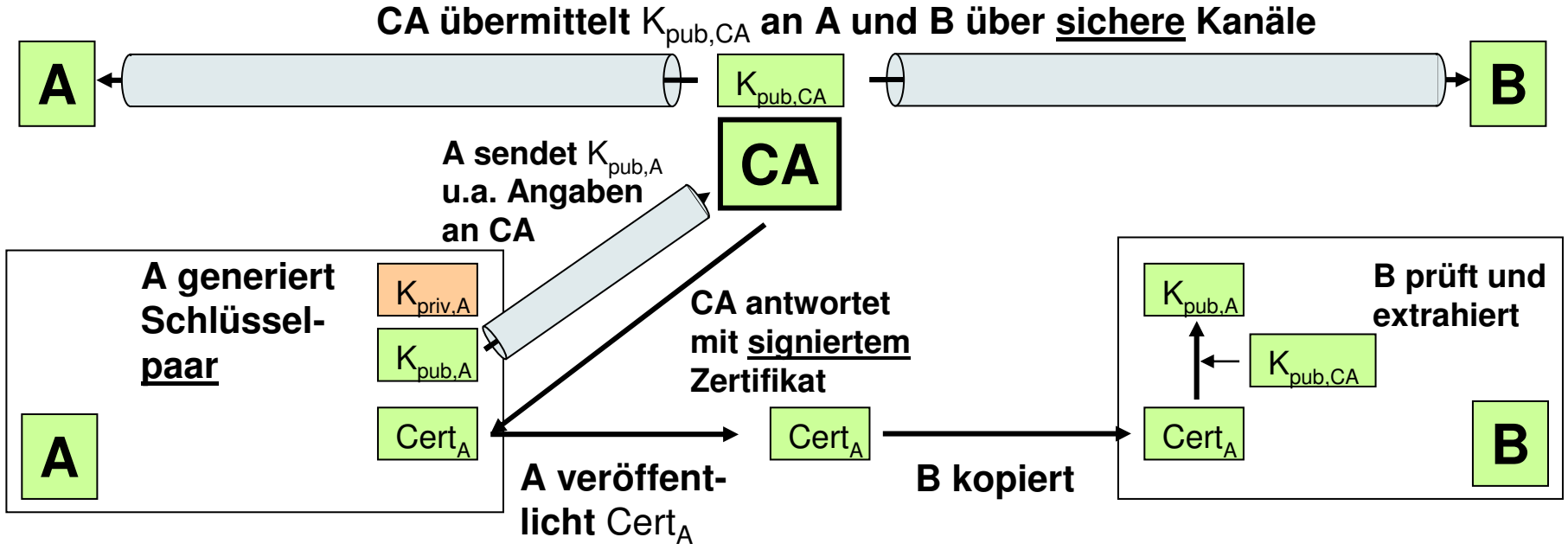
(kann oft & einfach stattfinden)

Zertifikate

- Problem: Öffentliche Schüssel können gefälscht werden!



- Lösung: Zertifizierung durch vertrauenswürdige Stelle



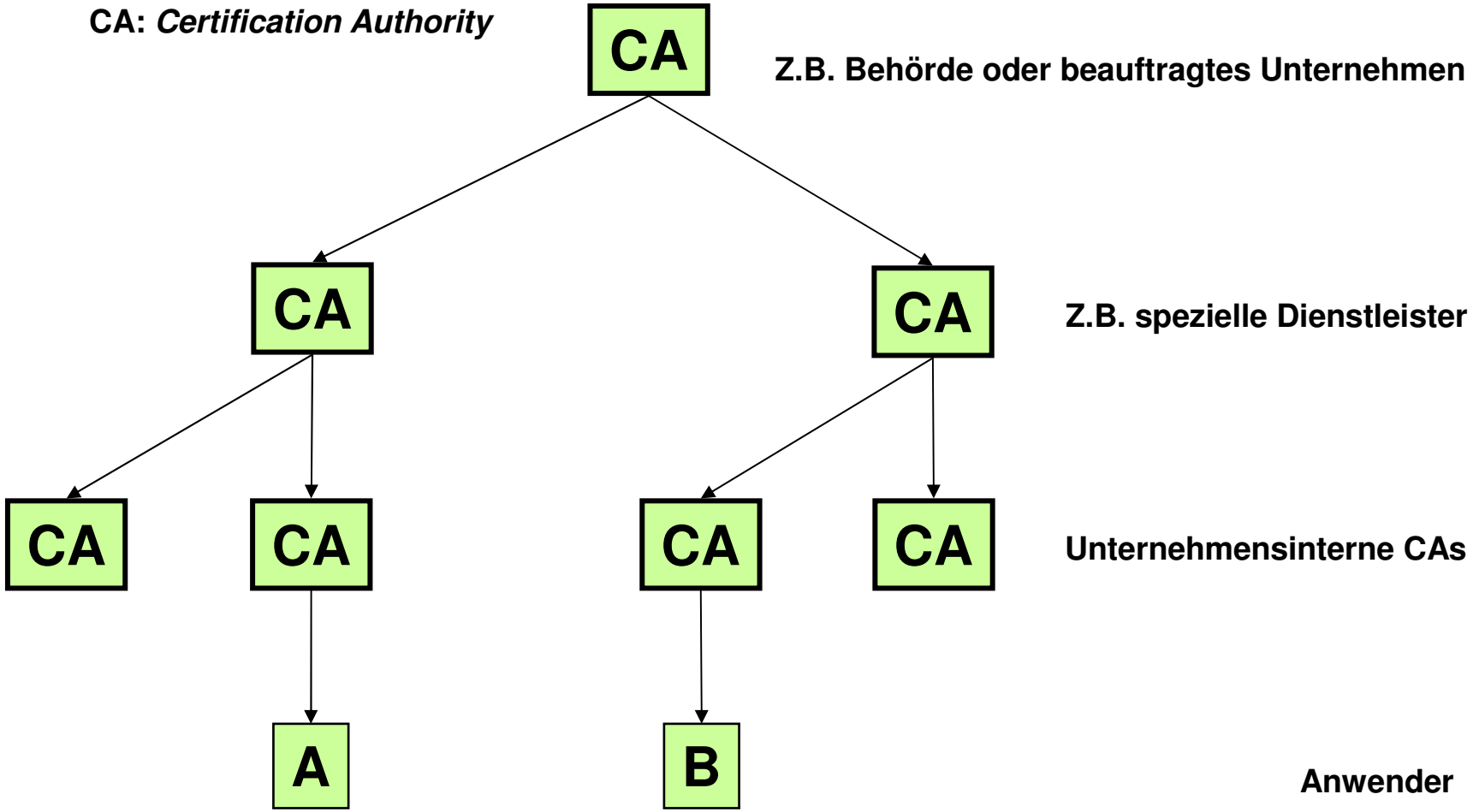
Zertifikate

- Inhalte eines Zertifikats
  - Öffentlicher Schlüssel
  - Name oder eindeutige Bezeichnung des Ausstellers
  - Name oder eindeutige Bezeichnung und weitere Daten des Eigentümers
  - Gültigkeitsdauer
  - Regeln und Verfahren zur Anwendung
  - Zulässige Anwendungs- und Geltungsbereiche des Zertifikats

  - Schließlich: Digitale Signatur der CA über alle Inhalte

Public Key Infrastructure (PKI)

- Hierarchie von Zertifikaten und Zertifizierungsstellen



Probleme bei der Übertragung auf Web Services

- Hash- und Verschlüsselungsfunktionen arbeiten mit Bytefolgen, SOAP-Envelopes besitzen aber eine feste Struktur
  - Arbeiten auf Dateiebene ist nicht möglich!
  - Wie lassen sich Dokumententeile verschlüsseln bzw. signieren?
- XML-Daten lassen sich variieren, ohne dass ihre Bedeutung sich ändert. Beispiele:
  - Verschiedene Zeilenende-Konventionen
  - White space-Zeichen (wenn nicht signifikant) dürfen ergänzt/entfernt werden
  - Zeichenliterale vs. Zeichenreferenzen, CDATA-Sektionen vs. Zeichenweises „escaping“
- Hash- und Verschlüsselungsfunktionen ergeben aber andere Ergebnisse selbst bei Änderung nur eines Bits!
  - Kanonische Darstellung von XML-Daten!?
- Binärdaten aus der Verschlüsselung: Wie in XML darstellen?
  - Base64-Codierung!

Probleme bei der Übertragung auf Web Services

- Warum ist das Vorgehen bei HTML mittels SSL/TLS keine Lösung?
  - Verschlüsselung möglich, nicht aber digitale Signatur
  - Dialoge bei abgelaufenen Zertifikaten o.a. Störungen nicht möglich
  - Lösungen außerhalb des WS verändern den Gesamtprozess und erhöhen die Komplexität. Konzepte wie WSDL werden unterlaufen.
- Standards
  - X.509 ITU-Standard für Zertifikate
  - **XML-Signature** W3C-Standard für Digitale Signatur mit XML, 12. Februar 2002 Syntax and Processing http://www.w3.org/TR/xmldsig-core/
  - **XML-Encryption** W3C-Standard für Verschlüsselung mit XML,10. Dezember 2002 Syntax and Processing http://www.w3.org/TR/xmlenc-core/
  - XKMS 2.0 W3C-Standard für Schlüsselverwaltung, 28. Juni 2005 http://www.w3.org/TR/xkms2/
  - SAML 2.0 OASIS-Standard für den Austausch von Identitäts- und Sicherheitsdaten per XML. http://www.oasis-open.org/specs/index.php#samlv2.0
  - XACML 2.0 OASIS-Standard für Zugangskontrolldaten: …php#xacmlv2.0
  - **WS-Security** OASIS-Standard, der regelt, wie SOAP-Nachrichten gesichert werden können. http://www.oasis-open.org/specs/index.php#wssv1.0

XML Signature: Einfaches Beispiel, separate Signatur

```
[s01] <Signature Id="MyFirstSignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
[s02]   <SignedInfo>
[s03]     <CanonicalizationMethod
            Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
[s04]     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
[s05]     <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
[s06]       <Transforms>
[s07]         <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
[s08]       </Transforms>
[s09]       <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[s10]       <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
[s11]     </Reference>
[s12]   </SignedInfo>
[s13]   <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
[s14]   <KeyInfo>
[s15a]    <KeyValue>
[s15b]      <DSAKeyValue>
[s15c]        <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
[s15d]      </DSAKeyValue>
[s15e]    </KeyValue>
[s16]   </KeyInfo>
[s17] </Signature>
```

XML Signature: Einbettende Signatur

```
[s01] <Signature Id="MyFirstSignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
[s02]   <SignedInfo>
[s03]     <CanonicalizationMethod
            Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
[s04]     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
[s05]     <Reference URI="#myEmbeddedObj">
[s06]       <Transforms>
[s07]         <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
[s08]       </Transforms>
[s09]       <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[s10]       <DigestValue>...</DigestValue>
[s11]     </Reference>
[s12]   </SignedInfo>
[s13]   <SignatureValue>...</SignatureValue>
[s14]   ...
[s15a]  <Object Id="myEmbeddedObj">
[s15b]    <MyObject xmlns="urn:mynamespace">
[s15c]      ...
[s15d]    </MyObject>
[s15e]  </Object>
[s16] </Signature>
```

XML Signature: Eingebettete Signatur – gut für SOAP!

```
[s01] <MyDoc xmlns="urn:mynamespace">
[s02]   <Object Id="myObjToSign">
[s03]     ...
[s04]   </Object>
[s05]   <Signature Id="MyFirstSignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
[s06]     <SignedInfo>
[s07]       <CanonicalizationMethod
              Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
[s08]       <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
[s09]       <Reference URI="#myObjToSign">
[s10]       ...
[s11]   </Signature>
[s12] </MyDoc>
```

XML Signature: Anwendung auf SOAP

```
<env:Envelope xmlns:env=... >
  <env:Head>
    <Signature Id="MyFirstSignature"
            xmlns="http://www.w3.org/2000/09/xmldsig#">
      <SignedInfo>
        <Canonicalization MethodAlgorithm=
           "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
        <SignatureMethod Algorithm=
           "http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
        <Reference URI="#myObj">...</Reference>
     </Signature>
  </env:Head>
  <env:Body>
    <b:getCityNameByZIP xmlns:b="http://beispiel" Id="myObj"
                        env:encodingStyle="..."
      <b:PLZ xsi:type="xsd:unsigned">65197</b:PLZ>
    </b:getCityNameByZIP>
  </env:Body>
</env:Envelope>
```