TECHNISCHE UNIVERSITÄT DRESDEN

Vorlesung „Service and Cloud Computing“

9. Sicherheitsaspekte in service-orientierten Architekturen

Dr.-Ing. Iris Braun

Gliederung

- Sicherheitsaspekte, Gefahren
- Grundlegende Konzepte und Begriffe

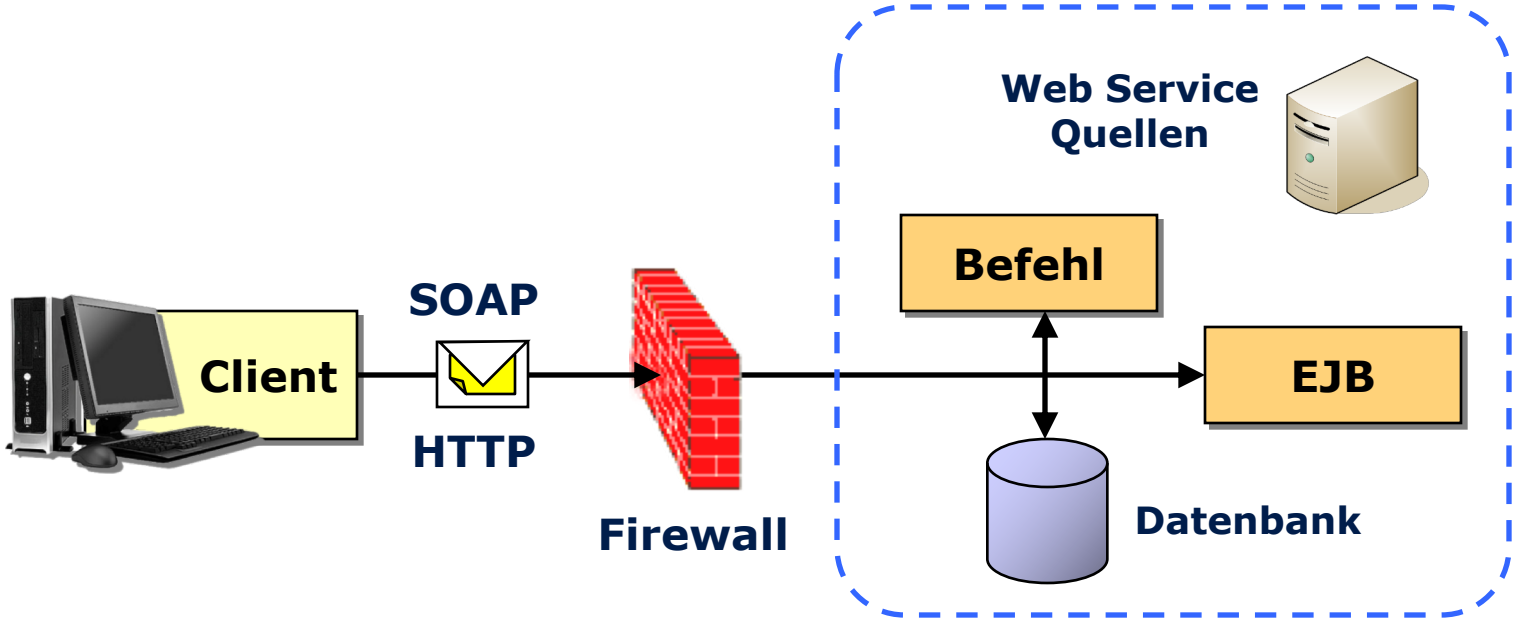
- Transportsicherheit
  - TLS, SSL

- Nachrichtensicherheit WS-Security
  - XML-Signaturen
  - XML-Verschlüsselung
  - Authentifizierung
  - Single-Sign-On (SAML)
  - Autorisierung (SAML)

- Sicherheit für REST-Services

Rückblick

- Bislang betrachtet: technische Möglichkeiten von Web Services zur Entwicklung einer SOA
  - Einfaches Anbieten von Diensten, kurze Entwicklungszeiten
  - Interoperabilität – Kooperation verschiedenartiger Systeme
  - Keine Firewall-Probleme bei SOAP oder REST über HTTP

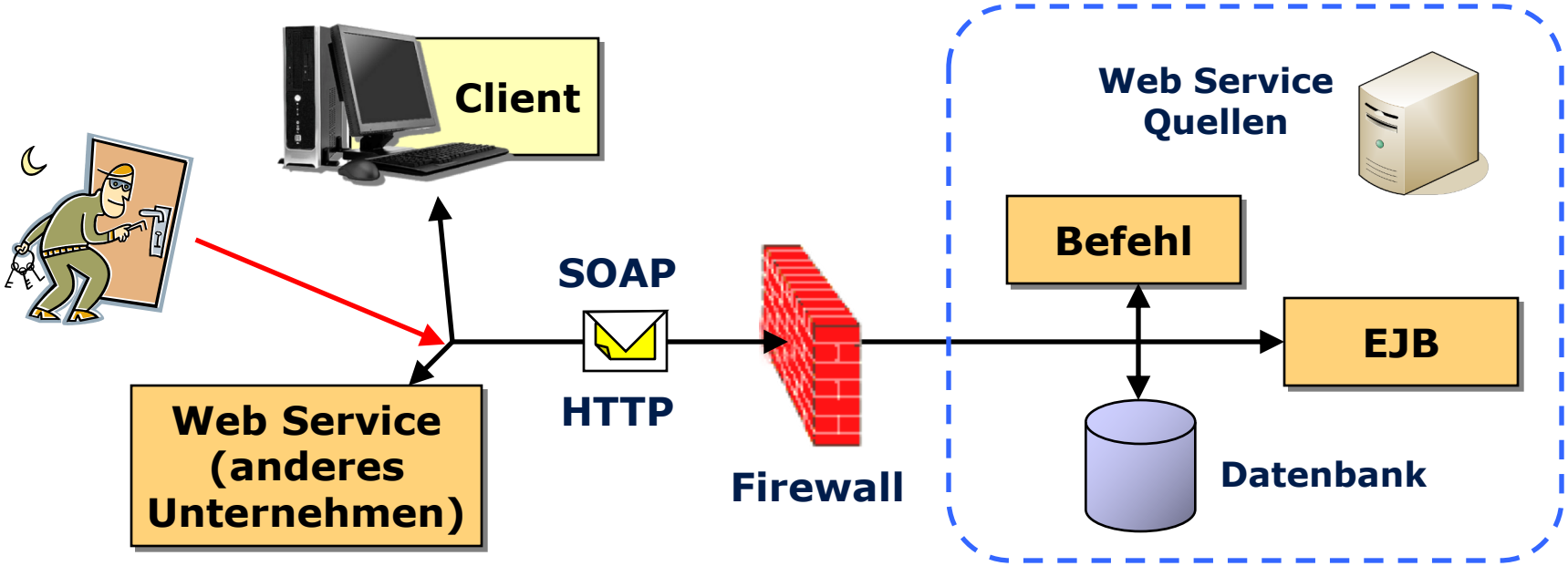


Problem

- Einsatz insbesondere innerhalb von (EAI) bzw. zwischen (B2B) Unternehmen interessant

⇒ Zahlreiche sicherheitsrelevante und kritische Daten

⇒ Risiko, das System durch die Einfachheit von Web Services Angriffen auszusetzen



Wo liegen grundsätzliche Probleme?

Von wem stammt die Nachricht?

- **Authentifizierung**: sichere Zuordnung einer Information zu ihrem Absender ⇒ Echtheit, Zuverlässigkeit, Glaubwürdigkeit

Was darf der Absender?

- **Autorisierung**: Zuweisung und Überprüfung von Zugriffsrechten auf Ressourcen

Wo liegen grundsätzliche Probleme? (2)

Wer kann eine Nachricht lesen?

- **Vertraulichkeit** (Konfidenz): Information ist nicht lesbar für unberechtigte Parteien

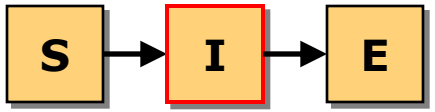
Wurde die Nachricht beim Versand verändert?

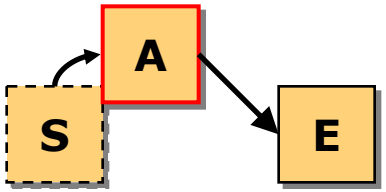
- **Integrität**: vollständige und unveränderte Datenübermittlung von tatsächlichem Absender zu Empfänger

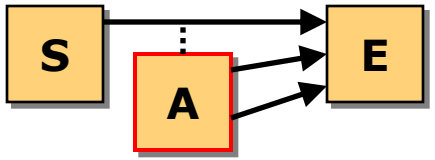
Welche Gefahren sind denkbar?

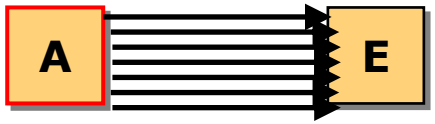
- **Abhören von Nachrichten**
  - SOAP-Nachrichten sind „plain text“ !
  - Auch Modifikation leicht möglich !
- **Man-in-the-Middle-Attacke**
  - Routing durch Intermediaries möglich
  - Intermediary könnte kompromitiert sein!
- **Spoofing**
  - Angreifer täuscht andere Identität vor
  - Oft Grundlage für andere Angriffe
- **Replay**
  - Mehrfaches Senden abgefangener Nachricht
- **Denial of Service (DoS)**
  - Überlastung des Empfängers durch sehr häufiges Senden

```
<soap:Envelope>
  <soap:Body>
    <user>ich</user>
    <pin>3456</pin>
  </soap:Body>
</soap:Envelope>
```









Abgrenzung

- Keine Betrachtung von
  - Schutz von Diensten gegen Angriffe durch Ausnutzung von Sicherheitslücken in der Implementierung
    - Exploits, blockierende Aufrufe, ...
  - Schutz der Infrastruktur
    - DoS-Attacken, Spoofing, ...
  - Privacy, Datenschutz

- Konzentration auf
  - Schutz der Daten (Nachrichten)
  - Schutz der Benutzer (Clients)

Sicherheitsaspekte

Bedeutungen von Sicherheit:

- Schutz vor Gefahren
- Gewissheit (Bestätigung einer Vermutung)
- Zuverlässigkeit (Eintreten des erwarteten Verhaltens)

Daraus folgen **Anforderungen für IT-Anwendungen**:

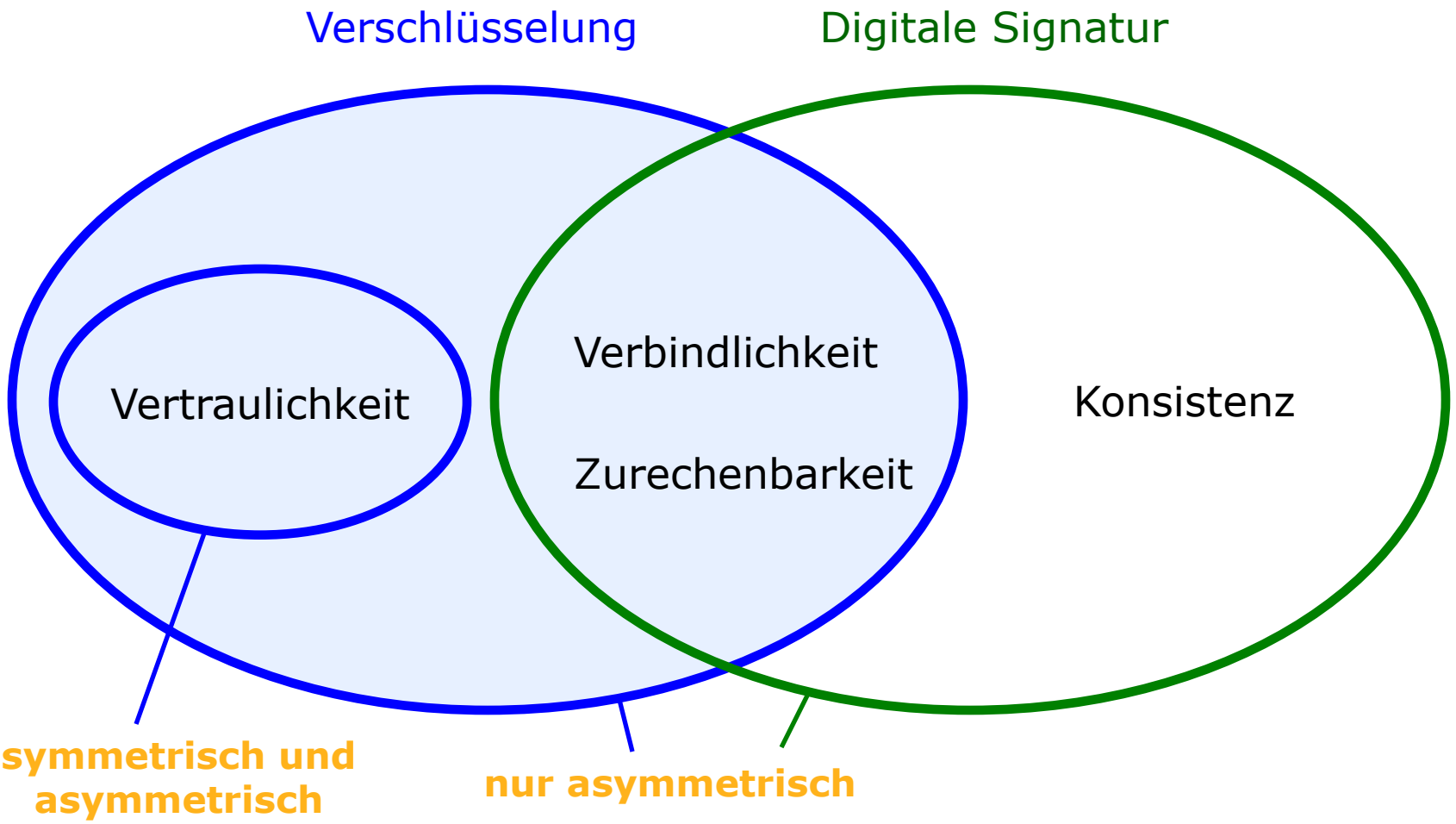
- *Vertraulichkeit:* Nachricht nur von adressiertem Empfänger lesbar
- *Berechtigung:* Dienstnutzer darf Dienst verwenden und Daten lesen, manipulieren, löschen
- *Integrität:* Nachricht wird beim Versand nicht modifiziert
- *Glaubwürdigkeit:* Absender einer Nachricht muss überprüfbar sein, Identität darf nicht vorgetäuscht werden
- *Verbindlichkeit:* Absender soll Versand einer Nachricht im Nachhinein nicht leugnen können

Technische Realisierbarkeit der Anforderungen

- 2 grundsätzliche Verfahrensklassen:
  - Kryptographie (modifiziert Nachrichtentext)
  - Signatur (bestätigt inhaltlich unveränderte Information)

Vertraulichkeit	Verschlüsselungsmechanismen: Transformation in für Dritte nicht lesbare Darstellung
Berechtigung	Digitaler Berechtigungsschein, Autorisierung
Daten-Konsistenz	Echtheitszertifikat: Beschreibung charakteristischer Eigenschaften des Originals, Abgleich (Konsistenzprüfung) beim Empfänger
Zurechenbarkeit	Digitale Unterschrift: Bestätigung der Urheberschaft und/oder Zustimmung zum Dokumentinhalt
Konsistenz	

Überblick


Verschlüsselung
Digitale Signatur
Vertraulichkeit
Verbindlichkeit
Zurechenbarkeit
Konsistenz
symmetrisch und asymmetrisch
nur asymmetrisch

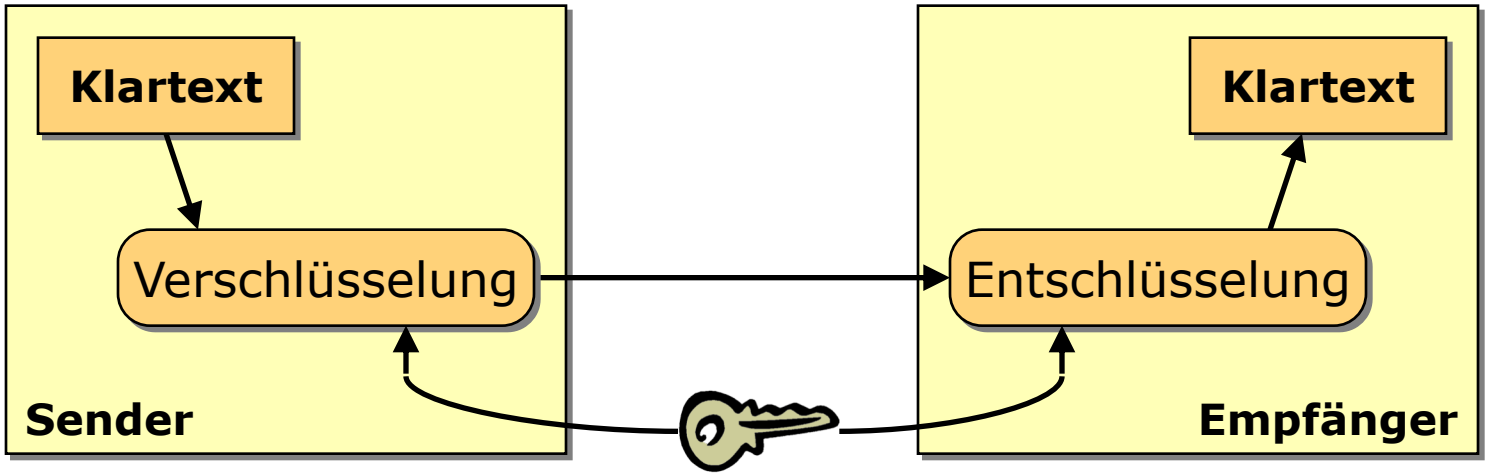
Verschlüsselung: symmetrisch

Symmetrische kryptographische Verfahren:

- Verwendung des gleichen Schlüssels für Chiffrierung und Dechiffrierung des Nachrichtentextes
- Vorteil: sehr schnell
- Nachteil: vorher Schlüsselaustausch notwendig

⇒ Keine Alternative für Web-Service-Anwendungen

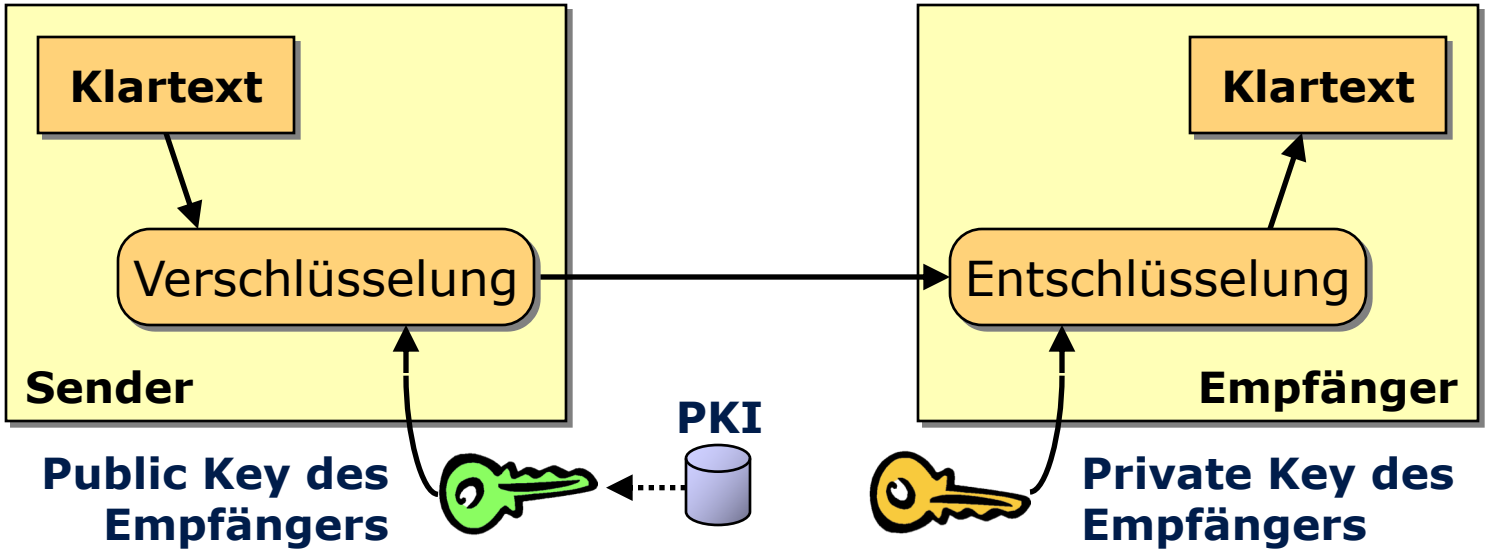
- Verfahren: Rijndael (AES), Twofish, 3DES



Verschlüsselung: asymmetrisch

Asymmetrische kryptographische Verfahren (public key):

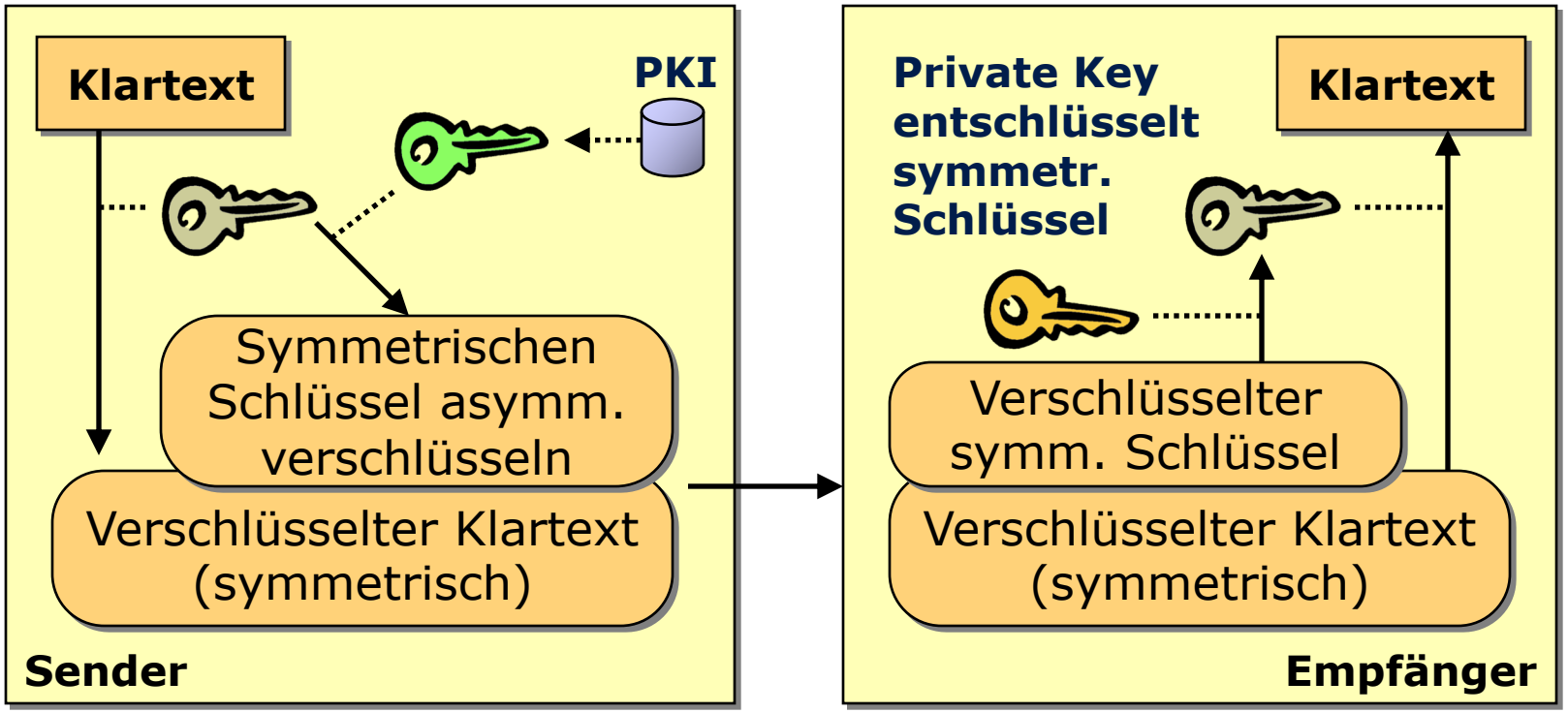
- Verschiedene Schlüssel für Ver- und Entschlüsselung
- Privater Schlüssel darf ohne Zusatzinformation nicht aus dem dazu passenden öffentlichen Schlüssel zu berechnen sein
- Vorteile: Geheimnis klein (pro Nutzer nur 1 privater Schlüssel), geringeres Schlüsselverteilungsproblem, aber langsam! (x1000)



Verschlüsselung: hybrid

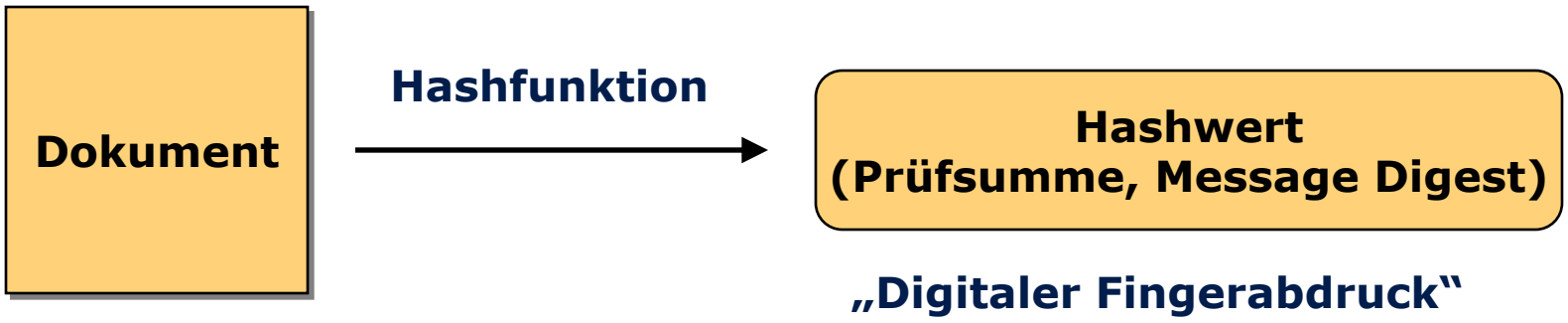
Hybride Verschlüsselung (schnell und sicher):

- Kombination asymmetrischer und symmetrischer Verfahren
- Bekanntes Verfahren: RSA (z.B. in Pretty Good Privacy, PGP)

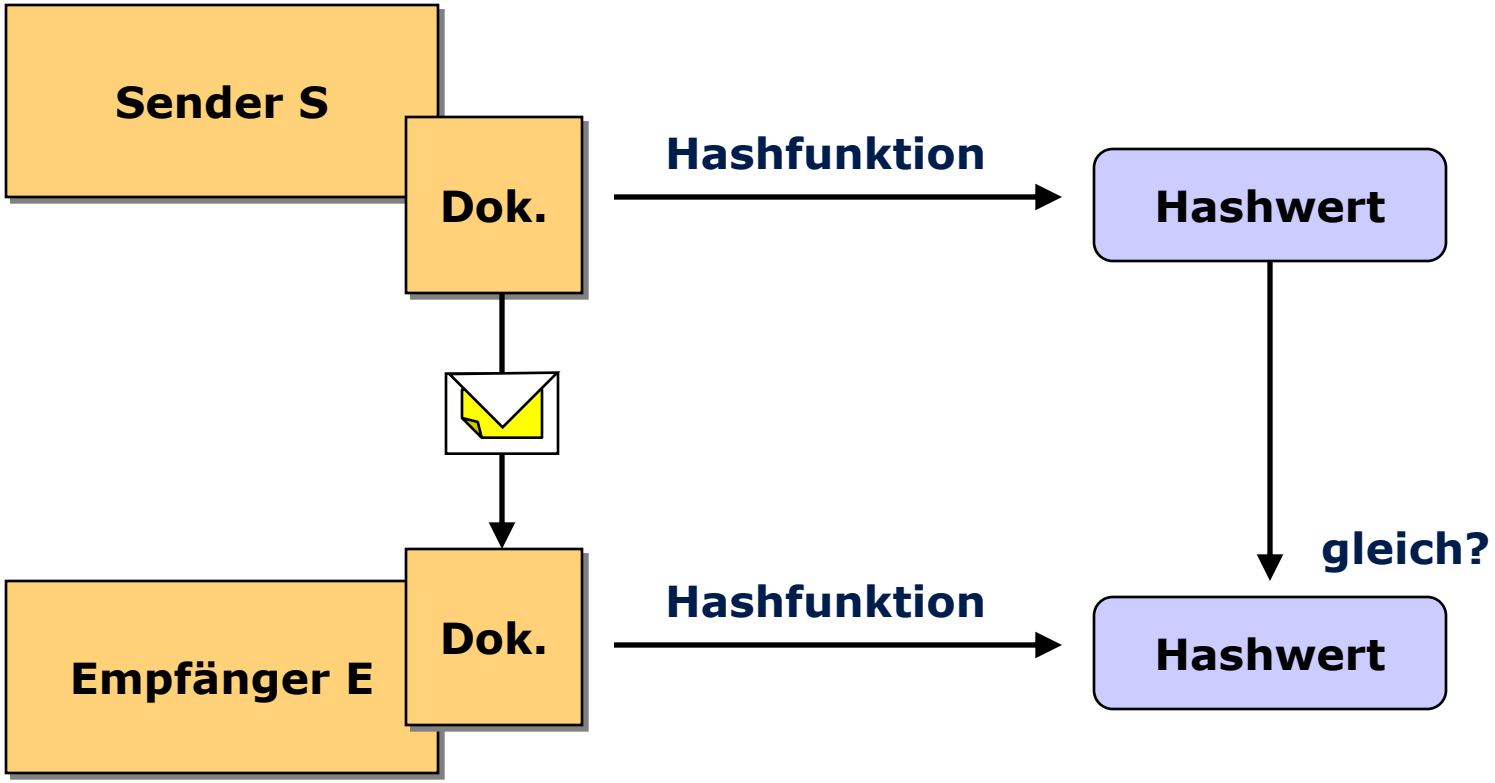


Digitale Signaturen

- Einsatzzweck: Daten, die bei Veränderung erhebliche Auswirkungen auf Anwendung hätten, Zurechenbarkeit
- Kann zusammen mit Verschlüsselung erfolgen (üblich: erst Signierung, dann Verschlüsselung)
- Basiert auf **Hashfunktion**: „One way function" (asymmetrisch)
  - leicht berechenbar, erzeugt Zeichenfolge bestimmter Länge
  - dazu inverse Funktion unbestimmbar
  - Änderung an Dokument soll anderen Hashwert erzeugen

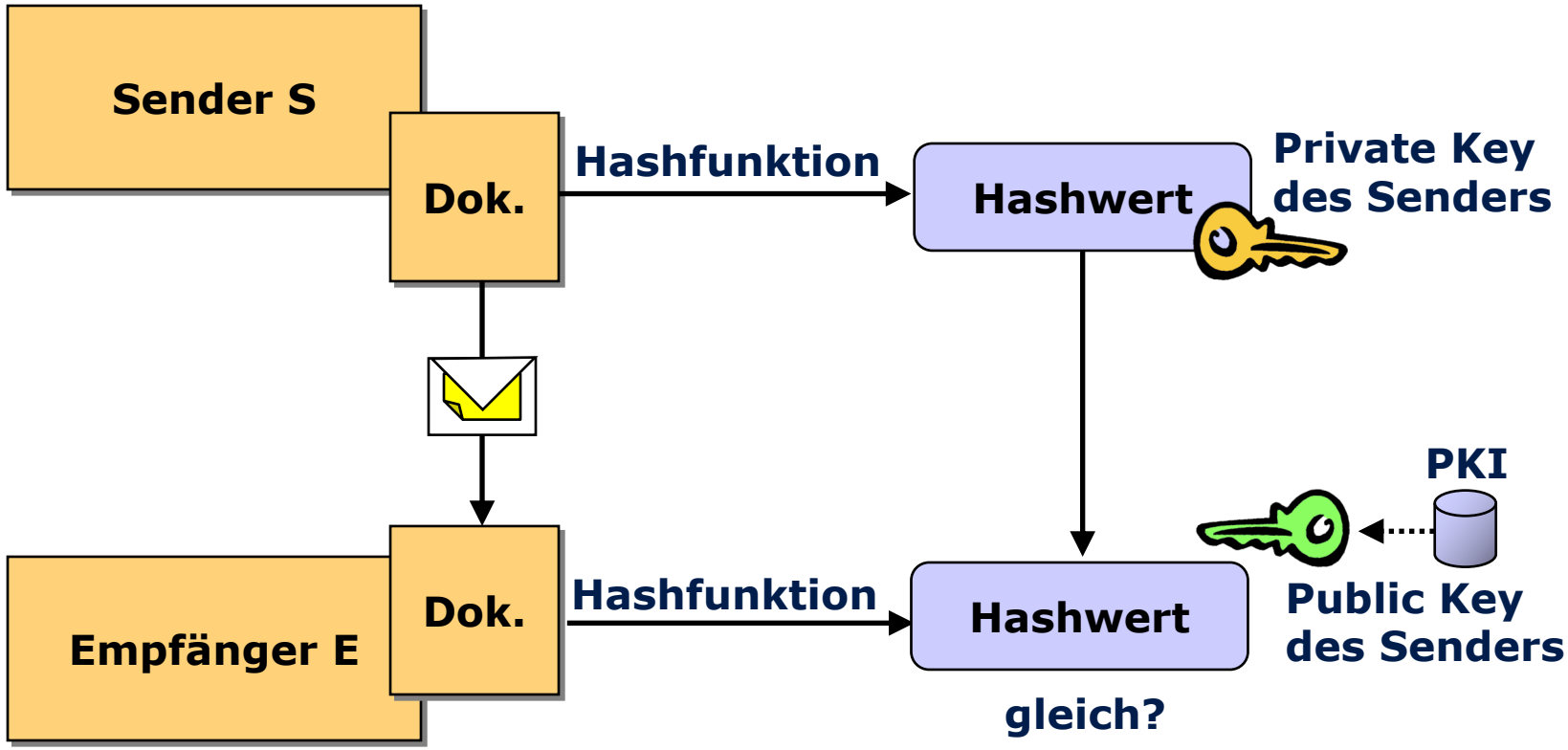


Digitale Signaturen: Beispiel Hashfunktion



Ziel: Zusicherung der Integrität – Änderungen am Dokument erkennen

Digitale Signatur: berechnet aus Message Digest und privatem Schlüssel des Unterzeichners

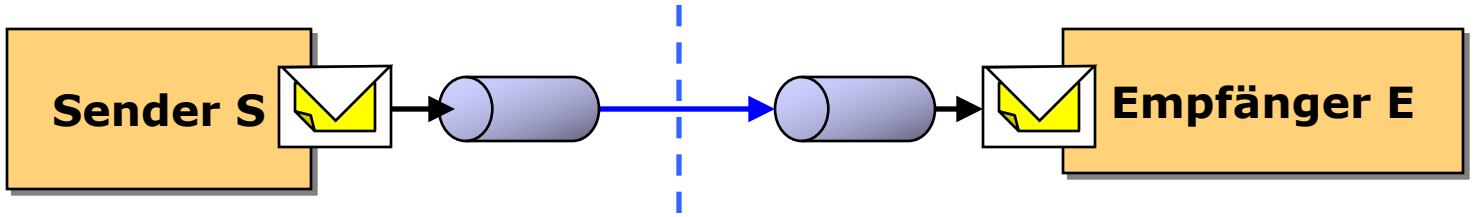


Transportsicherheit

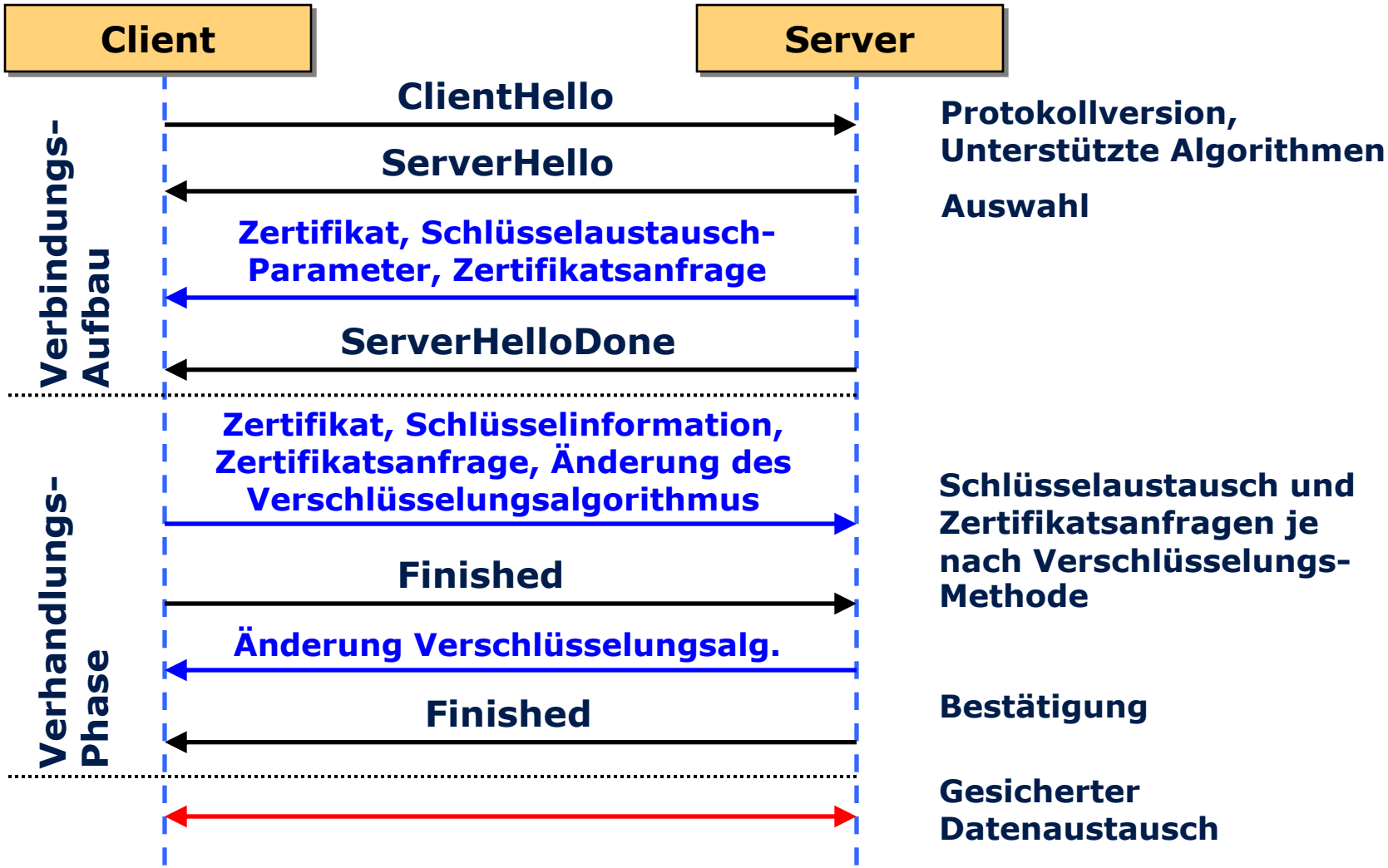
- Gewährleistung meist durch mehrere Schichten des ISO/OSI-Schichtenmodells
- Ziel: Nachrichten nicht abhör- und veränderbar
- Auf Vermittlungsschicht: IPsec, meist in Virtual Private Networks

Transport Layer Security (TLS):

- Weiterentwicklung von Secure Socket Layer (SSL)
- Anwendungsunabhängiges Protokoll, verbindungsorientiert
- Transparent: Transport verschlüsselt, auf Empfängerseite nach Empfang im Klartext vorhanden
- Vergleichbar mit Rohrpostsystem:



Ablauf: TLS/SSL-Verbindungsaufbau



Punkt-zu-Punkt-Verbindung

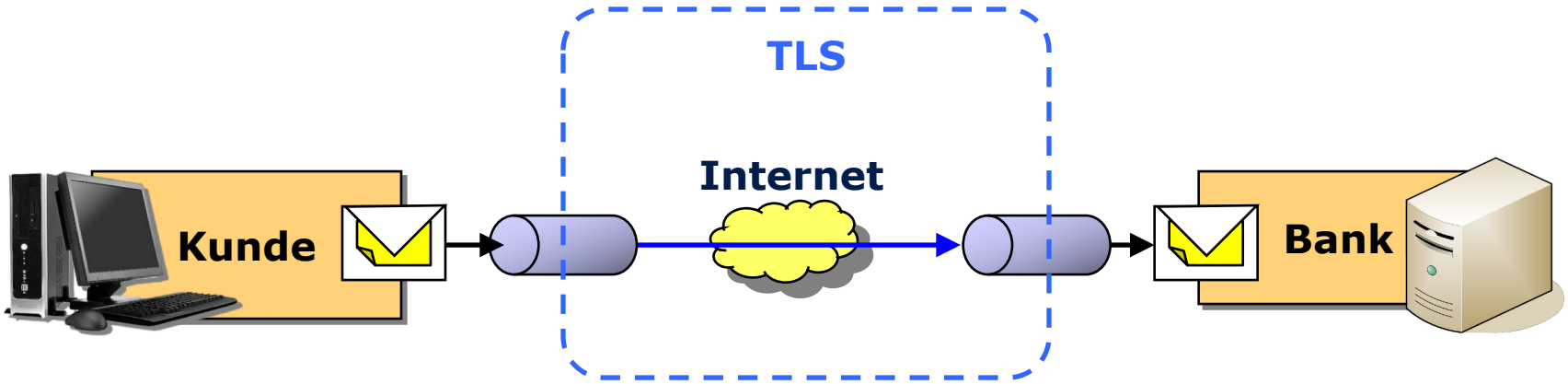
Einsatzgebiet von TLS:

- Kommunikation zwischen genau 2 Teilnehmern

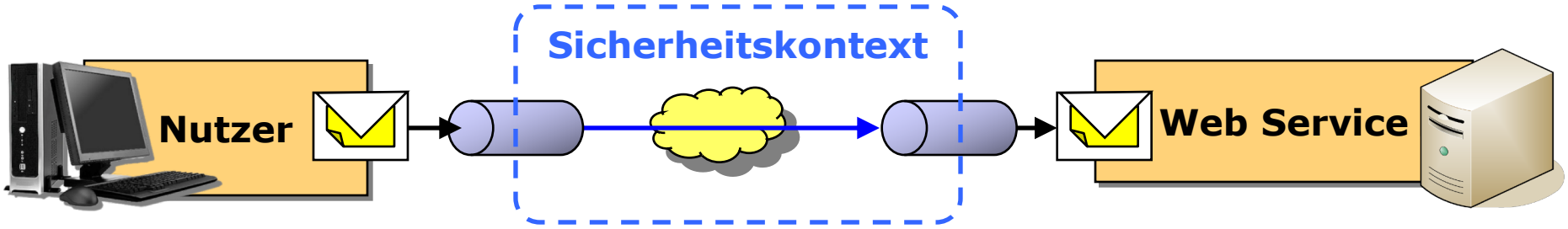
⇒ Echter Transportkanal zwischen Sender und Empfänger

- Beispiele:
  - Onlinebanking: genau ein Kunde ⇔ eine Bank
  - Web-Seiten: Browser-Client ⇔ Web-Server

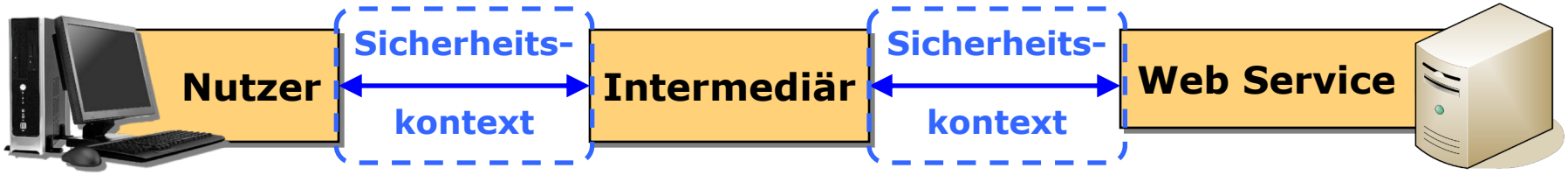
⇒ Für eine echte SOA mit Web Services geeignet?



Sicherheitskontext



Bei Punkt-zu-Punkt-Verbindung von TLS/SSL:



Wie sichert man Ende-zu-Ende-Kommunikation von Web Services?

Nachteile von TLS/SSL

- Aufbrechen des Sicherheitskontextes bei komplexer Web-Service-Kommunikation mit Zwischenstationen (Intermediären)
- Nur vollständige Verschlüsselung
  - keine feingranulare Sicherung von Nachrichten (Teilelemente)
- SSL-Handshake nicht in SOAP-Aufruf integriert
- Aufwand für Beschaffung von Server-Zertifikaten
  - Strategie für zentrale Zertifikat-Ablage notwendig
- Langsamer SSL-Handshake vor jedem SOAP-Aufruf
- Asynchrone Aufrufe durch Handshake nicht möglich
- Kein Absenderbeweis

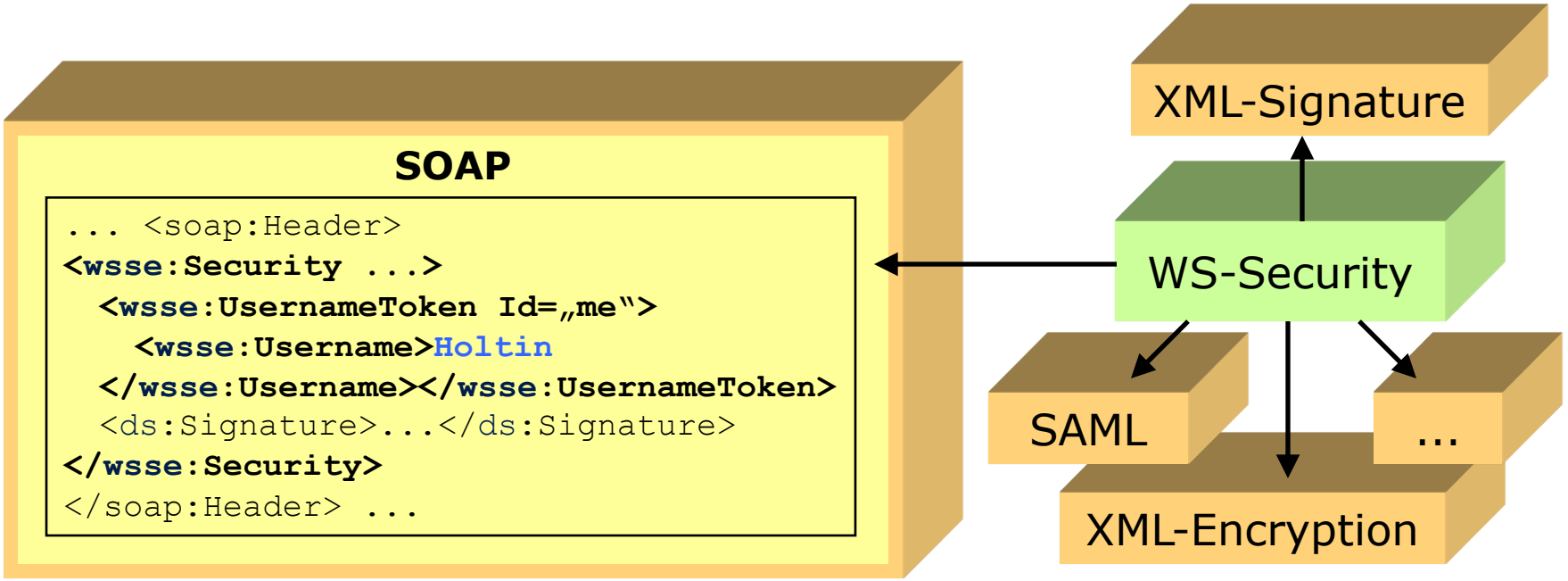
- **Vorteil:** erprobt und effektiv, Hardwarebeschleunigung verfügbar

WS-Security (OASIS-Standard)

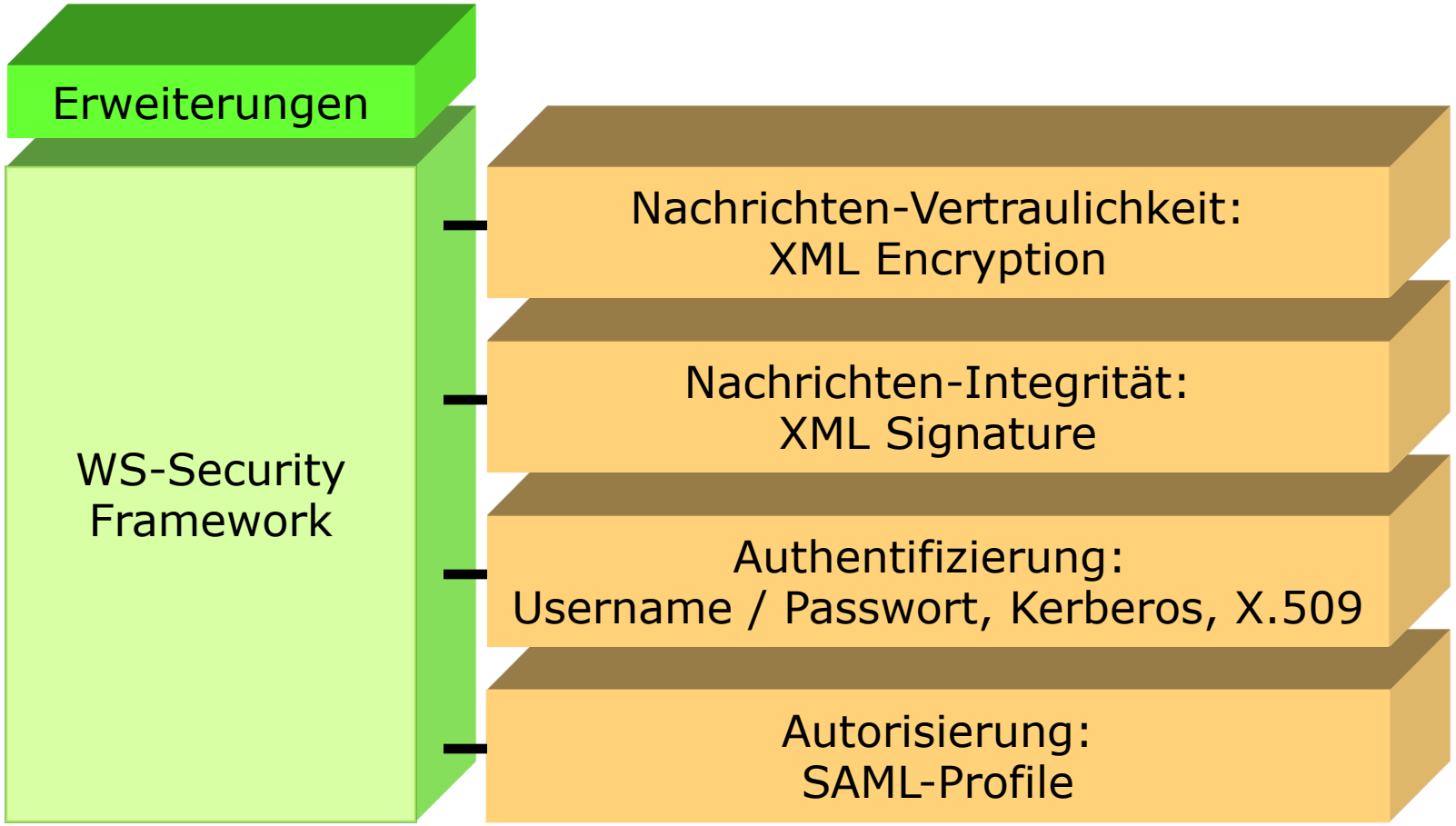
- Entwicklung von Microsoft, IBM und Verisign, 2002
- Erweiterung von SOAP ermöglicht Einsatz von XML-Signatur / XML-Verschlüsselung, PKI und anderen Standards

⇒ Standardisierter Einsatz bestehender bewährter Verfahren

- Dient auch als Basis für weitere WS-Sicherheitsstandards



WS-Security Framework



Nachrichtensicherheit bei Web Services

- XML-Dokumente
  - Trennung von Inhalt und Beschreibung (Elemente+Attrib.)
- Verschlüsselung/Signierung von Teilen des Dokuments möglich
  - z.B. nur Inhalt eines Elements
  - Folge: meist nicht schema-erhaltend (z.B. Boolean-Werte), trotzdem gültiges, wohlgeformtes XML-Dokument

```
<soap:Envelope>
  <soap:Body>
    <teil1>
      <da>true</da>
      <wert>1</wert>
    </teil1>
    <teil2>tag</teil2>
  </soap:Body>
</soap:Envelope>
```

```
<soap:Envelope>
  <soap:Body>
    <teil1>
      (verschlüsselt)
    </teil1>
    <teil2>tag</teil2>
  </soap:Body>
</soap:Envelope>
```

XML-Signaturen

- W3C-Standard: XML-Signature Syntax and Processing
- Signatur bezieht sich in einem XML-Dokument immer auf das Startelement eines Teilbaums
- SOAP: Signierung von Elementen im Body, weil nur dort tatsächlich Nutzinformation liegen sollte
- Problem: XML „formatfrei" – „white spaces" erlaubt, XML-Prozessoren dürfen solche Formatierungsanteile aber weglassen!

Offensichtlich unterschiedlich
⇒ Verschiedene Signaturen
Aber gleiche Information!

```
<soap:Envelope><soap:Body>
<teil1><da>true</da><wert>1</wert>
</teil1><teil2>tag</teil2>
</soap:Body></soap:Envelope>
```

```
<soap:Envelope>
  <soap:Body>
    <teil1>
      <da>true</da>
      <wert>1</wert>
    </teil1>
    <teil2>tag</teil2>
  </soap:Body>
</soap:Envelope>
```

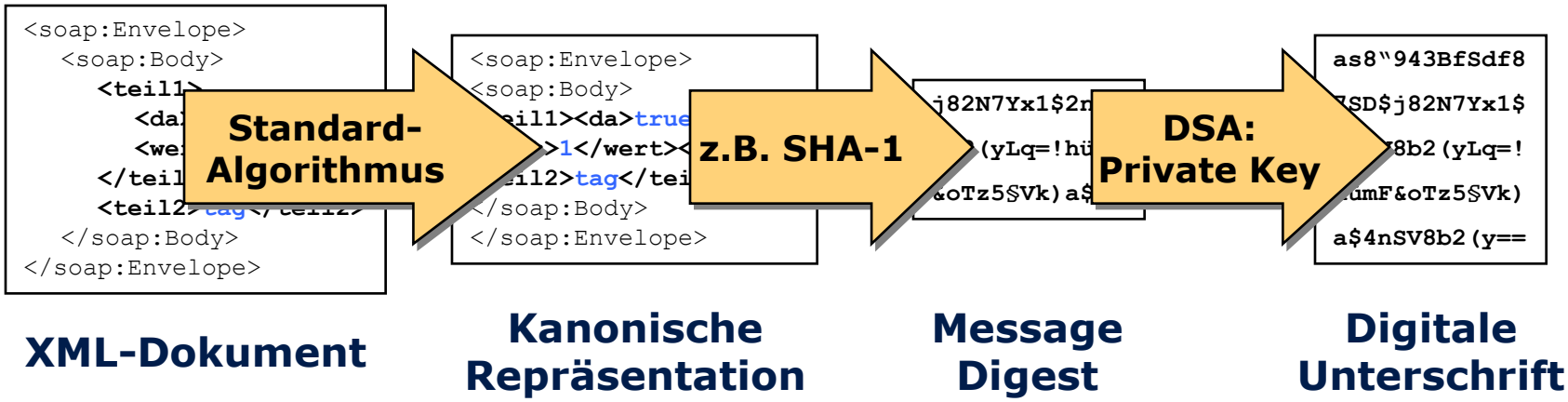
XML-Signaturen

- XML-konforme Umformatierung während des Transports möglich
- Lösung: mehrdeutiges XML-Dokument wird in eindeutige Normal-form gebracht, aus der die Signatur berechnet wird

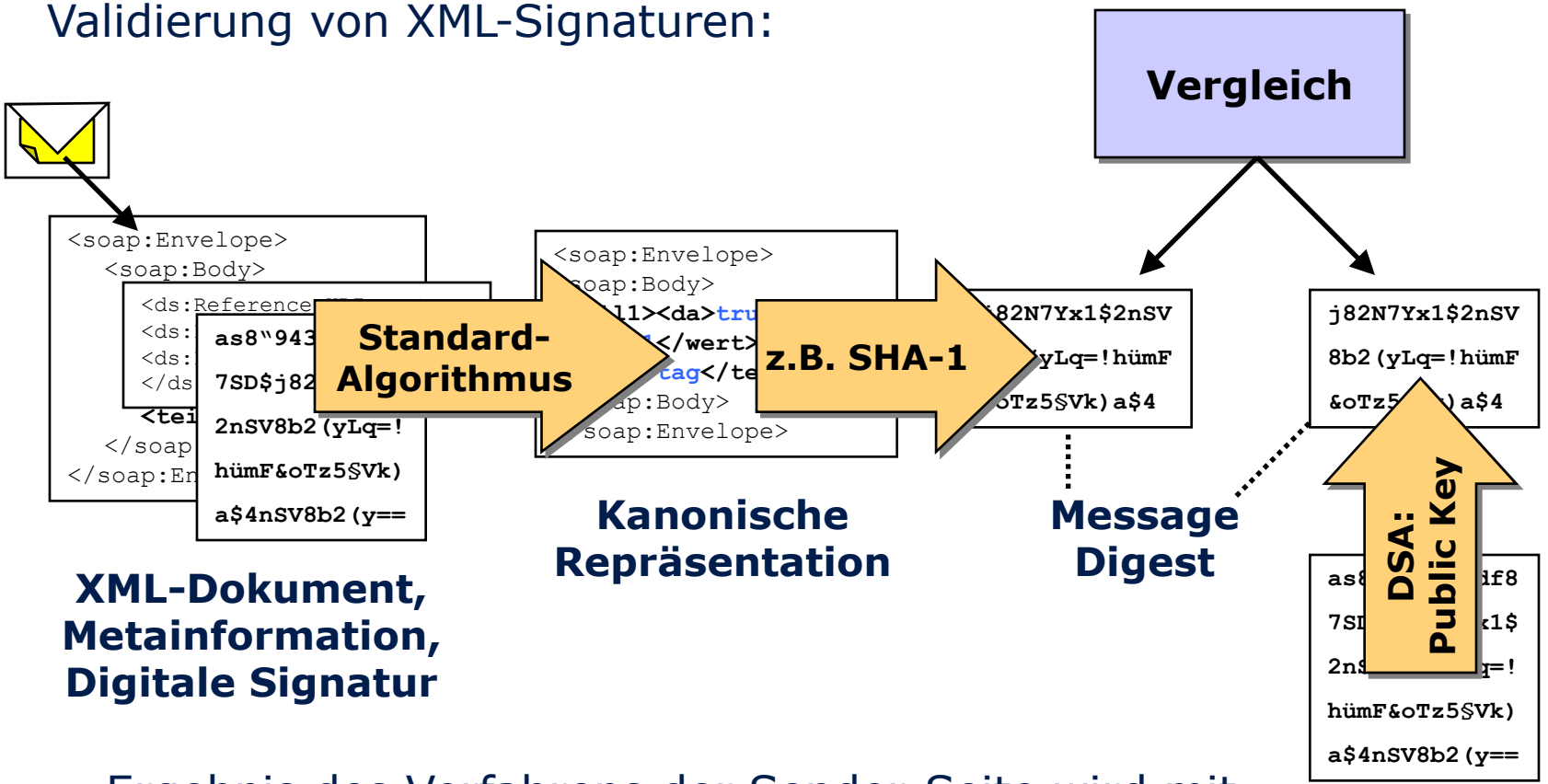
⇒ Canonical XML (W3C-Definition)

- Fingerabdruck: Message Digest wird mit SHA-1 berechnet
- Signatur: berechnet aus Digest und Schlüssel des Unterzeichners

Erzeugung:



Validierung von XML-Signaturen:



⇒ Ergebnis des Verfahrens der Sender-Seite wird mit aus digitaler Unterschrift gewonnenem Digest vergl.

XML-Signaturen

Bei asymmetrischen Verfahren:

- Originalnachricht oder Prüfsumme lässt sich auf Empfängerseite mit dem öffentlichen Schlüssel des Senders wiederherstellen

- Arbeiten der XML-Signature Working Group abgeschlossen, Referenzimplementierungen verfügbar
- Zentrale Schlüsselverwaltung (Public Key Infrastructure, PKI) für öffentliche Schlüssel empfehlenswert

- Ziel: Überprüfbarkeit von Urheberschaft einer Nachricht und deren Konsistenz, keine Vertraulichkeit!

XML-Encryption

- Ebenfalls W3C-Standard
- Ziel: Vertraulichkeit sichern
- Verschlüsselung von XML-Teilbäumen sowie selektive Verschlüsselung von Elementinhalten ohne Elementtags möglich

Grundlegende Vorgehensweise:

- Notwendige Randbedingungen definieren
  - Verschlüsselungsverfahren, Initialisierungsparameter
  - Verschlüsselungsgranularität (elementweise, ...), Schlüssel
- Generierung des für Dritte nicht mehr lesbaren Dokuments aus dem Eingangsdokument

XML-Encryption

Spezifikation definiert

- Vokabular zur Beschreibung der zur Entschlüsselung notwendigen Informationen
- Grundlegende Verarbeitungs-anweisungen

```
<soap:Envelope>
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <Inhalt>
      <da>true</da>
      <wert>1</wert>
    </Inhalt>
  </soap:Body>
</soap:Envelope>
```

```
<soap:Envelope>
  <soap:Header>
    <wsse:Security>
      <xenc:ReferenceList>...
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <xenc:EncryptedData Id=...>
      <EncryptionMethod ...> ...
      <ds:KeyInfo ...> ...
      <xenc:CipherData>
        <xenc:CipherValue>...</>
      <xenc:CipherData>
    </xenc:EncryptedData>
  </soap:Body>
</soap:Envelope>
```

Schemaerhaltende Verschlüsselung

- Ziel: Verschlüsselung eines XML-Dokuments in ein Dokument, das wiederum dem verwendeten Schema entspricht
- Voraussetzung:
  - Schema muss strukturell erhalten bleiben
  - Datentypen dürfen nur in erlaubtem Bereich liegen

⇒ Wenig erfolgreich – bei *int* 2^{16} = >4 Milliarden Alternativen, bei *bool* nur 2 (Gegenteil oder gleicher Wert), leicht zu erraten

⇒ Keine wirkungsvolle Sicherheit realisierbar

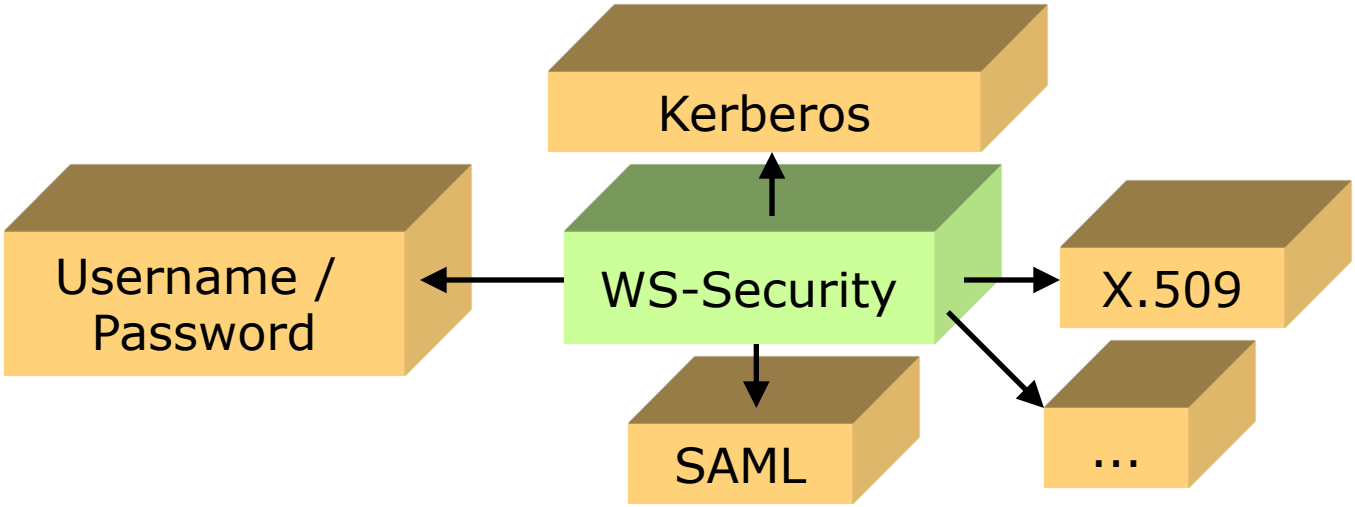
```
<soap:Envelope><soap:Body>
  <Bestellung>
    <erfolg>true</erfolg>
    <wert>16€</wert>
  </Bestellung>
</soap:Body></soap:Envelope>
```

```
<soap:Envelope>...<soap:Body>
  <Bestellung>
    <erfolg>xs:bool</erfolg>
    <wert>xs:string</wert>
  </Bestellung>
</soap:Body></soap:Envelope>
```

WS-Security: Authentifizierung

- Viele verschiedene Möglichkeiten, um Token (Identitätszeugnis) für Identitätsbeweise zu verwenden, z.B.
  - Benutzername / Passwort (Basic Authentication)
  - Kerberos-Ticket (verschlüsselt durch Aussteller, kann durch Dienstanbieter verifiziert und zusätzlich signiert werden)
- WS-Security erlaubt Verwendung beliebiger Token

⇒ Flexibilität, Erweiterbarkeit



WS-Security: Beispiel Authentifizierung

```
<soap:Envelope>
   <soap:Header>
     <wsse:Security xmlns:wsse="...">
      <wsse:UsernameToken>
        <wsse:Username>reisebuero</wsse:Username>
        <wsse:Password Type="wsse:PasswordDigest">
          1859023745
        </wsse:Password>
      </wsse:UsernameToken>
     </wsse:Security>
   </soap:Header>
   ...
</soap:Envelope>
```

⇒ **UsernameToken nur für Anwendungen mit Sicherheit auf der Transportschicht, z.B. HTTPS (HTTP mit SSL-Sicherung)**

SAML: Security Assertion Markup Language

- OASIS-Standard seit 2001
- Ziel: Verfahren zum Austausch sicherheitsrelevanter Informationen zur Authentifizierung und Autorisierung, standardisierte Beschreibung existierender Sicherheitsmodelle
- Vollständig programmiersprachen-/plattform-/herstellerunabh.

Bestandteile:

- ***Assertions***: Aussagen über Sitzungsinformationen
  Beispiel: „Der Nutzer verwendet ein Passwort, das gültig ist."
- Protokolle für Anfragen und Übermittlungen solcher Aussagen (Assertion Request/Response)
- Bindungen und Profile zur Integration von SAML in andere Spezifikationen (auch WS-Security)

SAML: Assertion-Typen

- **Authentication**
  - Bestätigung, dass der genannte Dienstnutzer zu einer bestimmten Zeit von einem bestimmten Vermittler authentifiziert wurde
- **Attribute**
  - Bestätigung, dass einem bestimmten Dienstnutzer statische oder dynamische Attribute wie Rollen oder Informationen zugeordnet sind
- **Authorization**
  - Erklärung einer Autorität, dass ein bestimmter Dienstnutzer die Erlaubnis zu einer Aktivität auf einer Ressource hat

-> Analogie zum Notar: Beglaubigung, Vollmacht

SAML: Eigenschaften von Assertions

- In XML beschrieben
- Verschachtelung wird unterstützt
  - Einzelne Assertion kann mehrere interne Aussagen enthalten
- Menge von Assertions kann zu SAML Profile zusammengefasst werden
- Von SAML Autorität ausgestellt
- Versionsnummerierung und Signierung möglich
  - digitale Signierung und Verschlüsselung

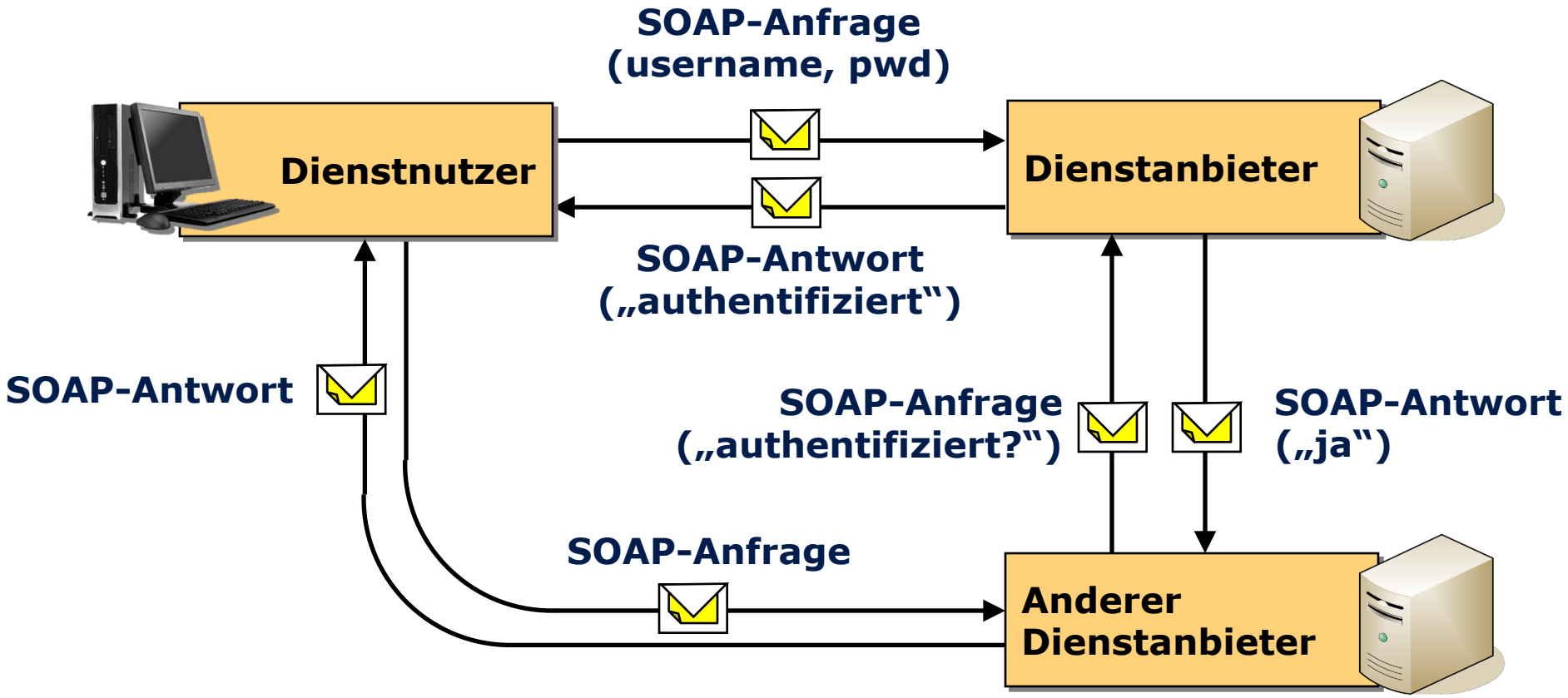
  $\Rightarrow$ Schutz gegen Replay-Attacken

SAML: Verwendung des Protokolls

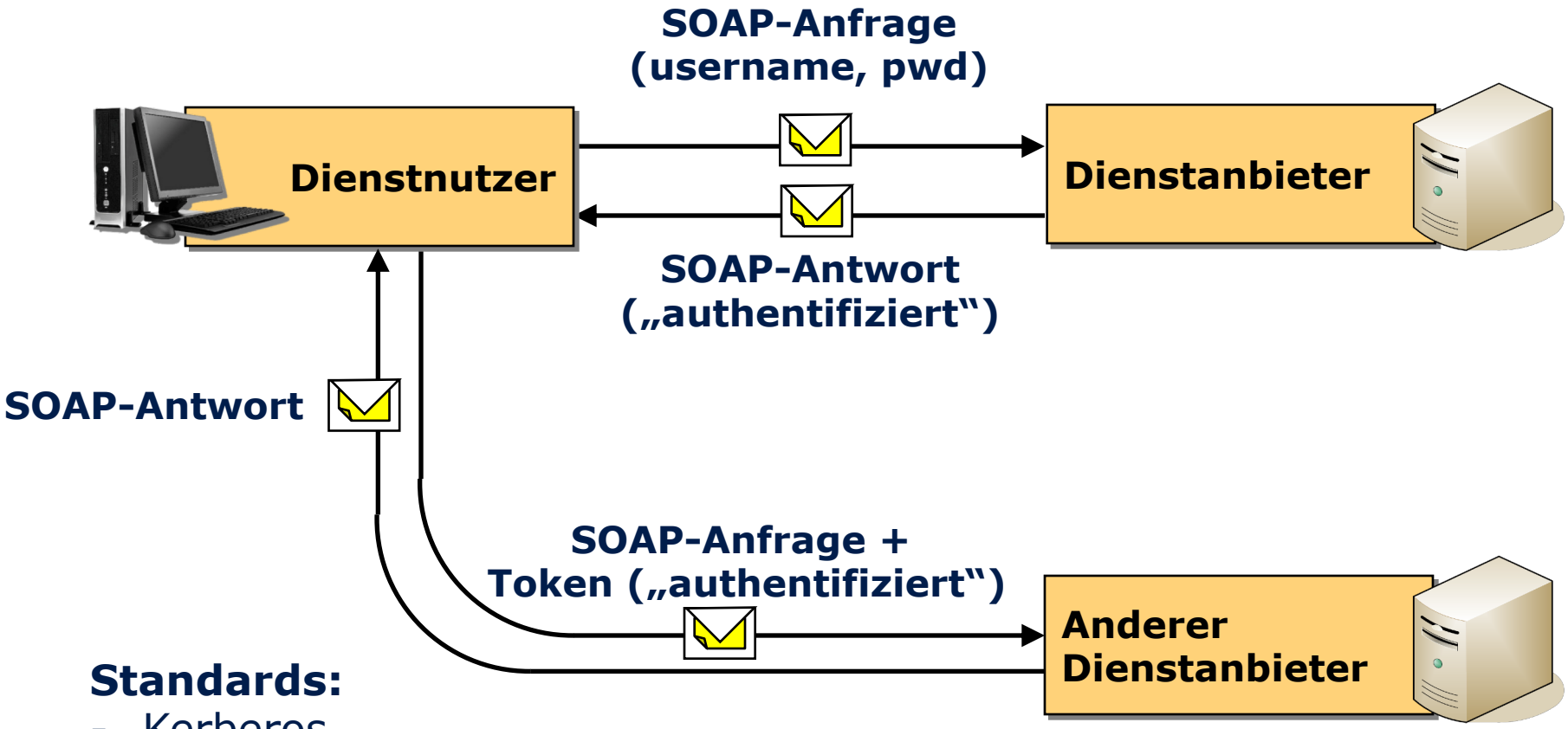
Verschiedene Anwendungsszenarien:

- **Single Sign-On** (ein Dienstnutzer ist nach der Anmeldung bei einem Dienst automatisch auch zur Benutzung von weiteren Diensten berechtigt)
- **Autorisierungsdienste** (eine „unbeteiligte" Autorität erzeugt und prüft Berechtigungen)
- **Verteilte Transaktionen** (mehrere Dienste sind gemeinsam an einer Transaktion beteiligt und teilen sich die Sicherheits-informationen)

Single Sign-on



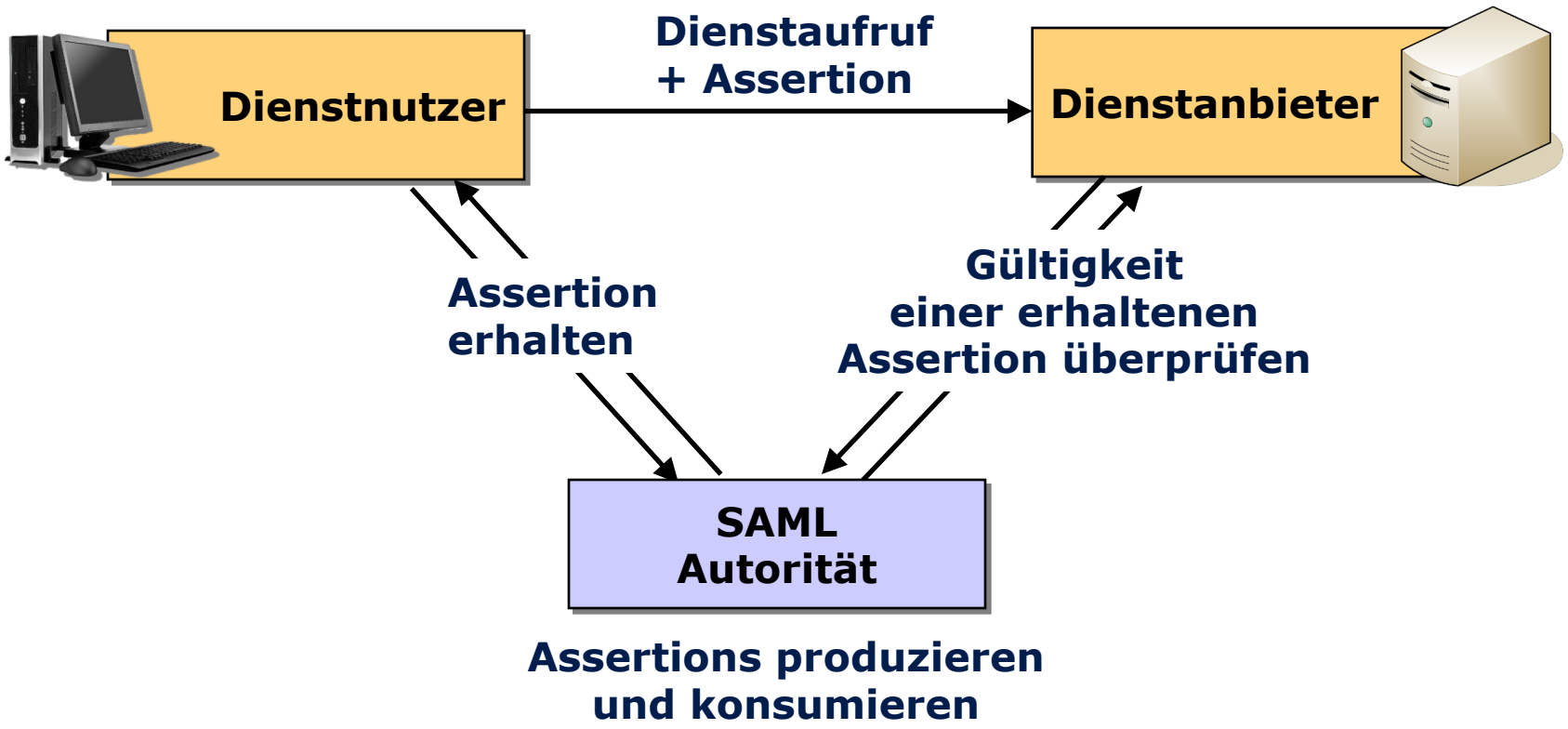
Single Sign-on



Standards:

- Kerberos
- OpenID Connect, OAuth2 (im REST-Umfeld und für Webseiten)
- .NET Passport/MS CardSpace/Windows Live ID (Webseiten)
- SAML, XACML (SOAP-basierte Web Services)

SAML: Autorisierungsdienst



SAML-Autorität ist keine zentrale Certificate Authority, Assertion kann von einem beliebigen Punkt im Netzwerk zugesichert werden

XACML und XKMS

XACML: eXtensible Access Control Markup Language

- XML-Sprache für Zugriffskontroll-Policies
- Request-Response-Protokoll zur Abfrage von Zugriffsrechten
- Regelwerk für die Beschreibung von Leserechten mit XML

XKMS: XML Key Management Specification

- W3C-Spezifikation
- XML-basierte Verwaltung einer PKI (Public Key Infrastructure)
- Menge von Web Services als Schnittstelle zu Key-Management-Systemen
- Unterstützung für die Verteilung bei Private/Public-Key-Mechanismen

WS-Security: Zusammenfassung

- Einsatz: hauptsächlich Sicherheit auf Nachrichtenebene
- Definiert, wie Teile der XML-Dokumente signiert oder verschlüsselt werden (neue Header-Elemente)
- Ermöglicht Verwendung (Sicherung) von
  - Verschlüsselung (Vertraulichkeit)
  - Signaturen (Integrität, Verbindlichkeit)
  - Authentifizierung und Autorisierung

  mittels verschiedener standardisierter Verfahren

⇒ Keine Alternative / Konkurrenz zu Standards wie XML-Signature, Encryption und SAML (Security Assertion Markup Language), sondern Verwendung dieser Standards

Sicherheit für REST-Services

HTTP-Authentication

- Basic Auth., Digest, NTLM (MS NT LAN Manager), SPNEGO, Kerberos
- Übertragung bei jedem Request notwendig (Zustandslosigkeit)

SSL/TLS

- für verschlüsselte Kommunikation zwischen Client und Server (Punkt-zu-Punkt!)
- Authentifikation des Services, Authentifikation des Clients (optional)

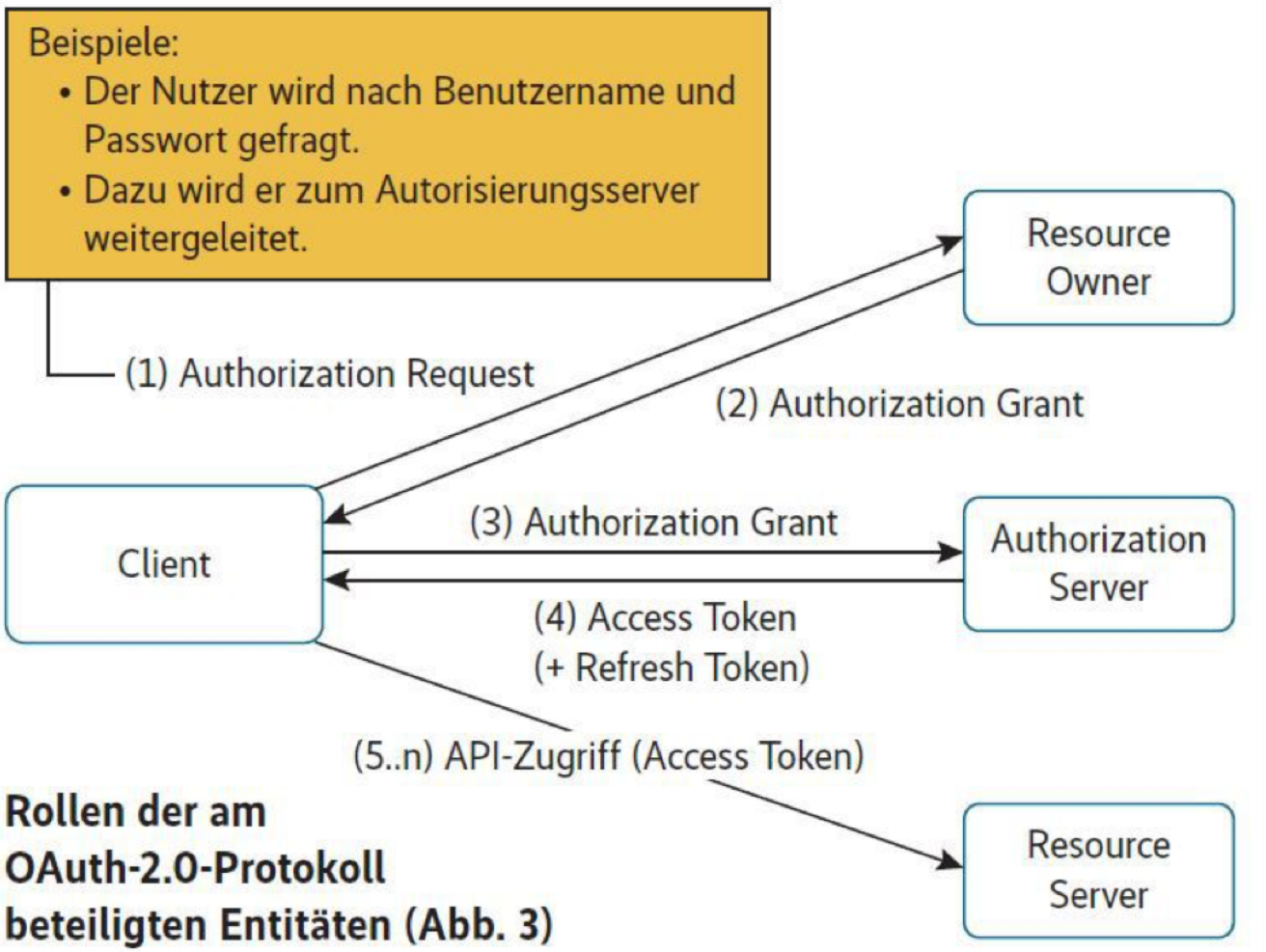
OpenID Connect

- ermöglicht Single-Sign-On
- Dezentrale Authentifizierung und Identitätsmanagement für Web-Apps
- Konzept der URL-basierten Identität

OAuth 2.0 (RFC 6749)

- Standardisiert von IETF OAuth Working Group
- unterstützt schwerpunktmäßig einen zentralisierten Autorisierungs- und Datenaustauschprozess sowie einen delegierten Aufruf von REST-Webservices
- Verwendet verschiedene Token-Formate u.a. JSON Web Token (JWT)

OAuth 2.0



Rollen der am OAuth-2.0-Protokoll beteiligten Entitäten (Abb. 3)

Quelle: http://www.heise.de/developer/artikel/Flexible-und-sichere-Internetdienste-mit-OAuth-2-0-2068404.html

JWT - JSON Web Tokens

JSON Web Tokens (RFC 7519)

- ermöglicht den Austausch von verifizierbaren **Claims** (Authentifizierungs- und Autorisierungsinformationen)
- besteht aus 3 Teilen: Header, Payload und Signatur
- werden mit Base64 kodiert
- JWT Token kann wie folgt aussehen:

```
jwt = base64(header) + "." + base64(payload) + "." +
base64(hash)
```

- JWT kann in der URL oder im HTTP-Header übertragen werden.

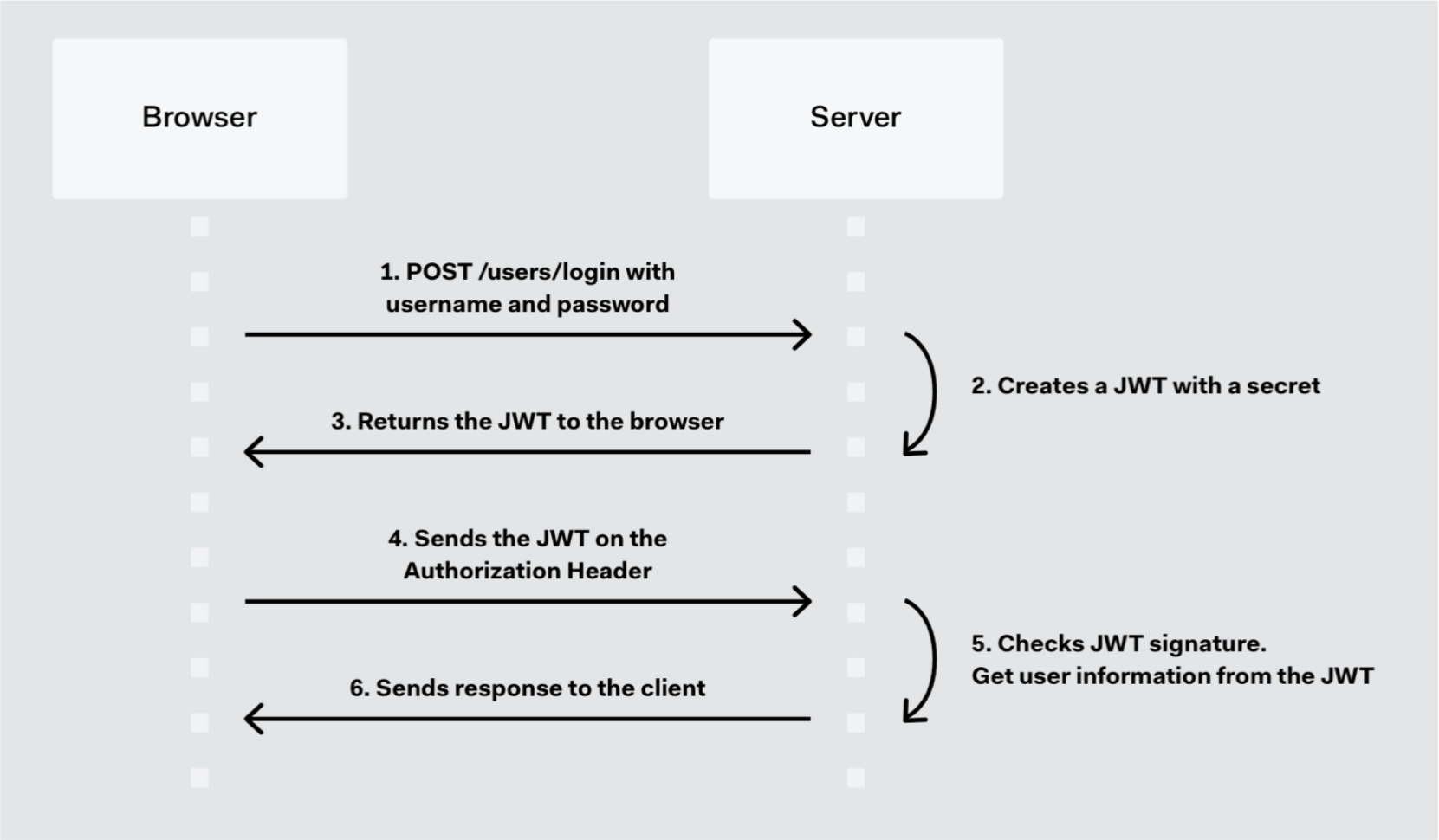
URL:`http://example.com/path?jwt_token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...`

HTTP-Header:

im Authorization-Feld als Bearer-Token: `Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...`

im Cookie-Feld: `Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...`

JWT – Ablauf



Bildquelle: https://auth0.com/learn/json-web-tokens/

Zusammenfassung Sicherheit

Transport Layer Security (TLS, SSL)

- Sicherung auf Transportebene
- nur für Punkt-zu-Punkt-Kommunikation, nicht bei Nachrichtenaustausch über Intermediäre geeignet (kein durchgängiger Sicherheitskontext)

WS-Security: Wichtigste Schutzziele auf Nachrichtenebene erreicht

- Vertraulichkeit durch Verschlüsselung mittels XML-Encryption
- Integrität und Verbindlichkeit mittels XML-Signature
- Authentifizierung (SAML)
- Autorisierung (SAML)

RESTful Services: eingeschränkte Sicherheitsmechanismen

- HTTPS für Verschlüsselung
- API-Schlüssel mit Keyed-Hash Message Authentication Code (HMAC) zur Authentifikation des Clients
- OAuth zusammen mit einem JSON Web Token (JWT) oder OpenId Connect für Authentifikation und Autorisierung

Lernaufgaben in AMCS

1) Was versteht man unter einem symmetrischen Kryptoverfahren?
   a) Sender und Empfänger nutzen jeweils unterschiedliche Schlüssel zur Ver- und Entschlüsselung der Nachricht.
   b) Empfänger und Sender müssen zeitgleich online sein, um die verschlüsselte Nachricht übertragen zu können.
   c) Sender und Empfänger verwenden den gleichen Schlüssel für die Ver- und Entschlüsselung.

2) Wie funktioniert die hybride Verschlüsselung?
   a) Der Sender verschlüsselt die Nachricht mit seinem privaten Schlüssel und sendet diese und seinen öffentlichen Schlüssel dann an den Empfänger.
   b) Der Sender verschlüsselt mit einem symmetrischen Schlüssel die Nachricht und dann den symmetrischen Schlüssel mit dem öffentlichen Schlüssel des Empfängers und sendet dies an den Empfänger. Dieser entschlüsselt erst den symm. Schlüssel mit seinem privaten Schlüssel und dann den Nachrichtentext mit dem symm. Schlüssel.
   c) Der Sender verschlüsselt mit dem öffentlichen Schlüssel des Empfänger und sendet die Nachricht und einen Hash-Wert an den Empfänger. Dieser entschlüsselt mit seinem privaten Schlüssel und bildet dann den Hash-Wert und vergleicht beide.

3) Was verbirgt sich hinter dem Begriff Authentifizierung?
   a) Zuweisung und Überprüfung von Zugriffsrechten
   b) Information ist nicht lesbar für unberechtigte Parteien
   c) sichere Zuordnung einer Information zu ihrem Absender
   d) vollständige und unveränderte Datenübermittlung