Schulung
Web Services
Sicherheit

Autor:
Thomas Bayer

predic8 GmbH
Moltkestr. 40
53173 Bonn

www.predic8.de
info@predic8.de

Version: 1.3

Inhalt

- Security, die Grundlagen
- Hash Funktionen, Digests
- Verschlüsselung
- Public Key Kryptographie
- Der X.509 Standard
- PKCS Standards
- Secure Sockets Layer SSL/TLS
- SSL/TLS mit Java und Apache Tomcat
- Java Security
- JEE und Servlet basierte Sicherheit
- XML Sicherheit
- XML Signature
- XML Encryption
- Web Service Sicherheit
- Der Web Services Security Standard
- WS-SecurityPolicy
- Single Sign On
- Federations
- Security Assertion Markup Language SAML
- SSO mit Shibboleth
- WS-Trust
- WS-Secure Conversation
- WS-Federation
- Single Sign On für das World Wide Web
- Kerberos
- Authorisierung

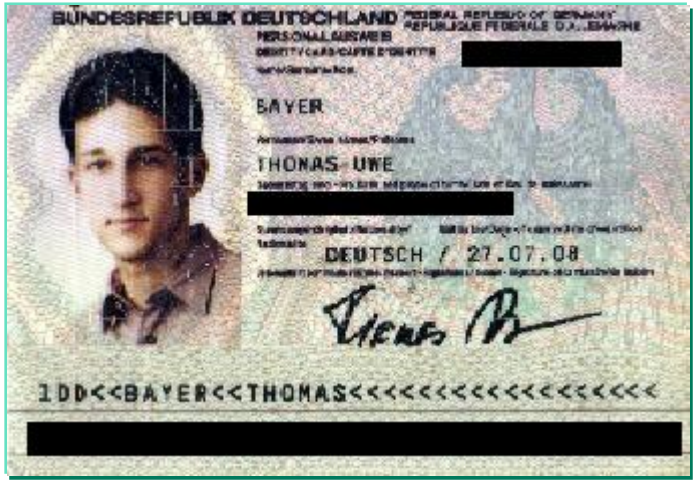
Security, die Grundlagen

Anforderungen an die Sicherheit

- Authentifizierung
  - Identität eines Users oder Programmes wird sichergestellt
- Authorisierung
  - Erlaubnis etwas zu tun
- Vertraulichkeit
  - Dokument kann nur vom rechtmässigen Empfänger gelesen werden
- Integrität
  - Dokument wurde bei der Zustellung nicht von Dritten verändert
- Non Repudiation
  - Sender kann nicht abstreiten, dass eine Nachricht von ihm stammt.

Identity

- Uniquely identifies a user or a program
- Consists of, e.g.:
  - Photo
  - Name
  - ID number



Authentication

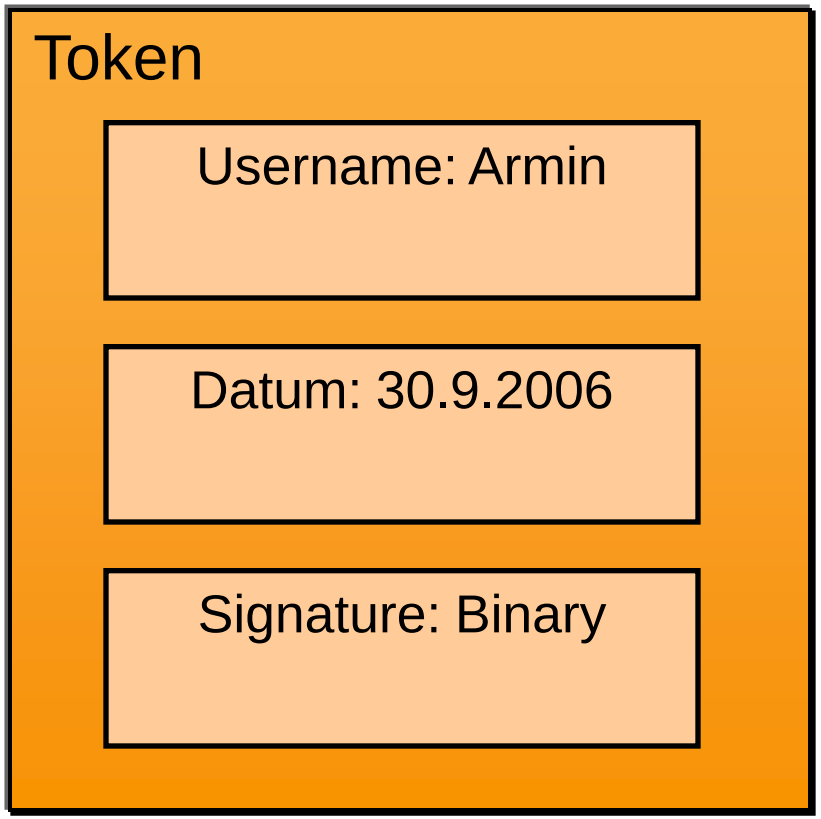
- Überprüfung, ob ein Prozess/Programm im Namen eines Principals handeln darf.
- Erfolgt durch die Überprüfung von Credentials (“Begleitpapieren”)
- Für die Authentifikation wird ein soziales Umfeld benötigt
  - Passämter, Staatsbürgerschaft, usw.

Claim

- Behauptung einer Entität z.B.
  - Username
  - Identität
  - Gruppenangehörigkeit
  - Recht
  - Fähigkeit
- Kann mit Credentials überprüft werden

Token

- Enthält ein oder mehrere Claims
- Kann signiert und verschlüsselt werden
- z.B. X509 Zertifikat (Signed Token)

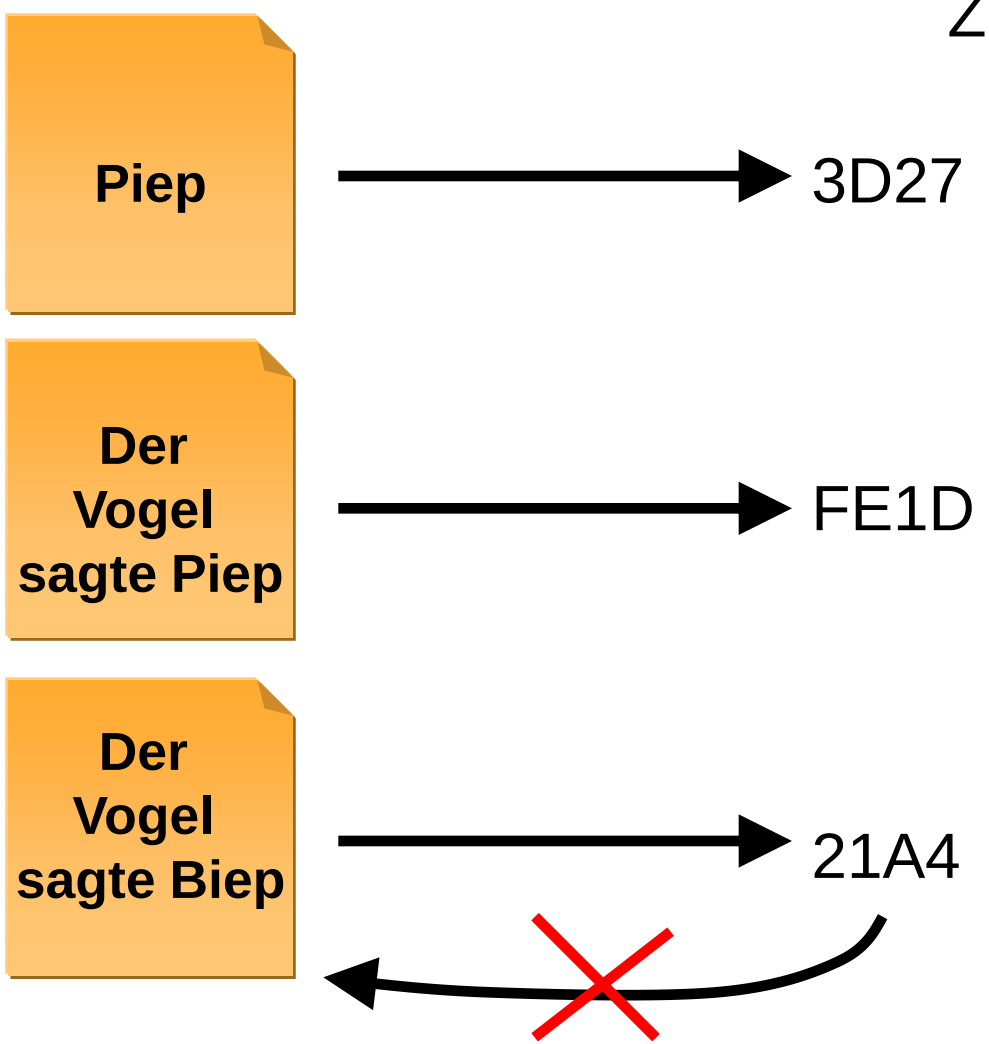


Hash Funktionen, Digests

Hash Funktion

predic8

Zerhacken und Mischen



Anwendungen für Hash Funktionen

- Prüfsumme
- Hash Tabellen
- Kryptographie
  - MD5
  - SHA-1
  - RIPE MD-128, RIPE MD-160
- Datei Identifikation für P2P Netzwerke

Kryptographische Eigenschaften von Hashs

- Preimage resistent

  Bei bekanntem h ist es schwer ein m zu finden, so dass gilt:

  h = hash(m)
- Second preimage resistent

  Bei bekanntem m1 ist es schwer ein m2 zu finden, so dass gilt:

  m1 ≠ m2

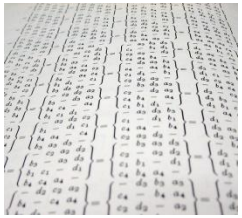
  hash(m1) = hash(m2)
- Collision-resistent

  Es ist schwer irgendein m1 und m2 zu finden, so dass gilt:

  m1 ≠ m2

  hash(m1) = hash(m2)

Hatte A bereits die Lösung?

1)  = 4722

2) hash(4722+11235) = 3F42D2

3) A sendet Aufgabe und 3F42D2 an B

4) B löst Aufgabe

5) A sendet Key 11235 an B

6) B prüft ob A bereits die Lösung hatte, als er die Aufgabe bekam

Passwort Speicherung

Users	
Joe	4D230F
Jack	E3D2FE
Sally	AB7325

Login: Jack

Password: sunrise

sunrise → Hash → E3D2FE

Message Digest Algorithm 5 MD5

- Weit verbreitet
- 128*Bit Länge = 32 Zeichen Hex Zahl
- Internet Standard RFC 1321
- Wurde 2005 kompromitiert
  - „Rainbow Tables“
- 1991 von Ronald Rivest um MD4 zu ersetzen
- Wird nicht mehr empfohlen

Secure Hash Algorithm SHA

- Entwickelt von der NSA
- Standards der US Regierung
- Familie von Algorithmen
  - SHA-1, SHA-224, SHA-256, SHA-384, SHA-512

SHA-1

- Wird verwendet von:
  TLS,SSL, PGP, SSH, S/MIME, IPsec
- Wurde als MD5 Nachfolger betrachtet
- Wurde kompromitiert
- 160-Bit Länge
- Max.Nachrichtenlänge: 2^64 -1 Bits

Angriff auf Hash Algorithmen

- Finden eines zweiten Dokumentes mit demselben Hashwert
- Kollisionen in SHA-1 Operationen:
  - Feb 2005: 1 in 2^69 Operationen
  - Aug 2005: 1 in 2^63 Operationen
  - Nov 2006: 1 in 2^52 Operationen

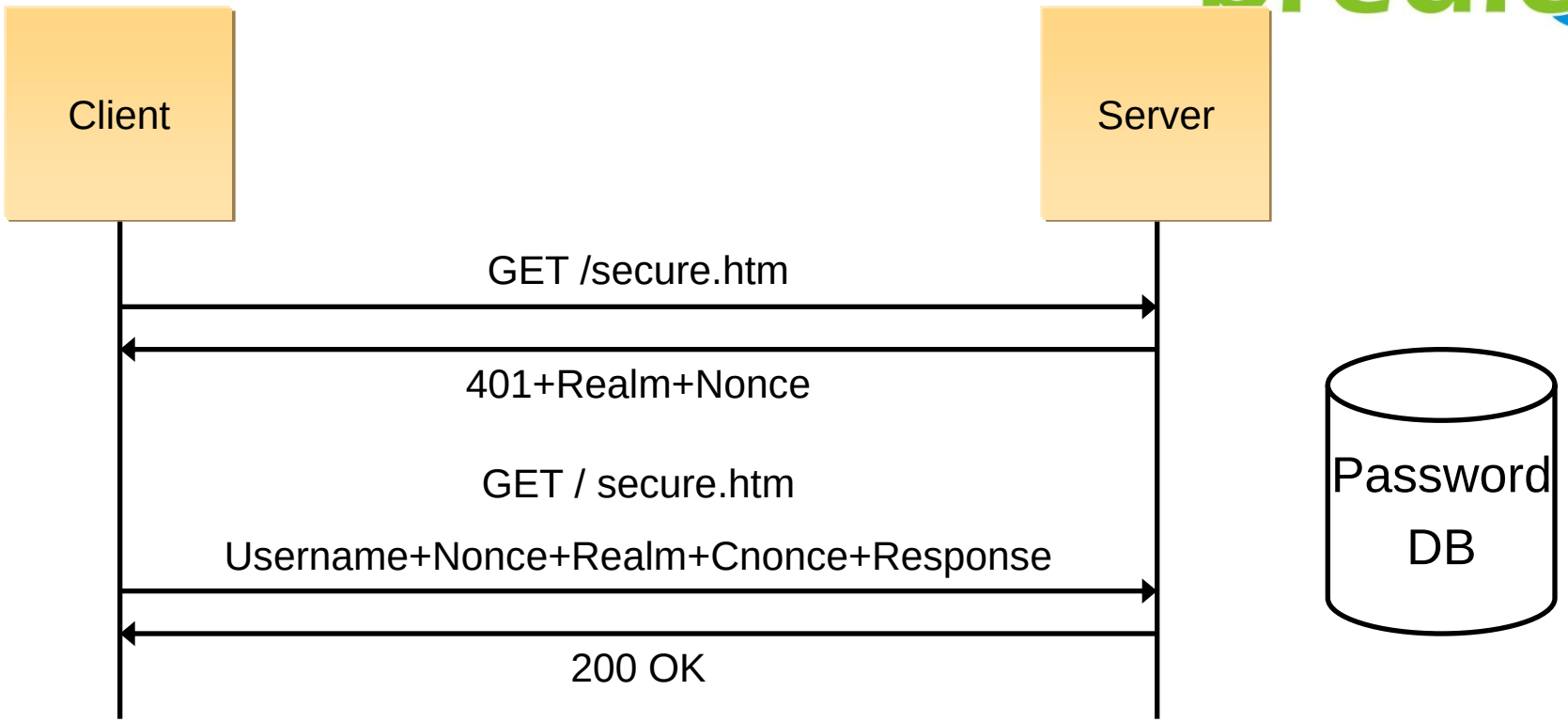
Cryptographic Nonce

- Wert, der nur einmal benutzt wird
- Wird meist zufällig erzeugt
- Schützt gegen Replay Attacks
- Beispiele
  - Timestamp, Zähler

HTTP Digest Access Authentication

- Verwendet MD5
- Passwort wird nicht übertragen

HTTP Digest Access Authentication



HA1 = md5 (username+Realm+Password)

HA2 = md5 (GET+/secure-htm)

Response = md5(HA1+nonce+rquestcounter+cnonce+qop+HA2)

Cnounce = ClientNonce

qop = Quality of Protection

Message Authentication Code MAC

- Für den Nachweis der Integrität von Informationen, die über ein unsicheres Medium übertragen wurden.
- Basiert auf einen gemeinsamen und geheimen Schlüssel

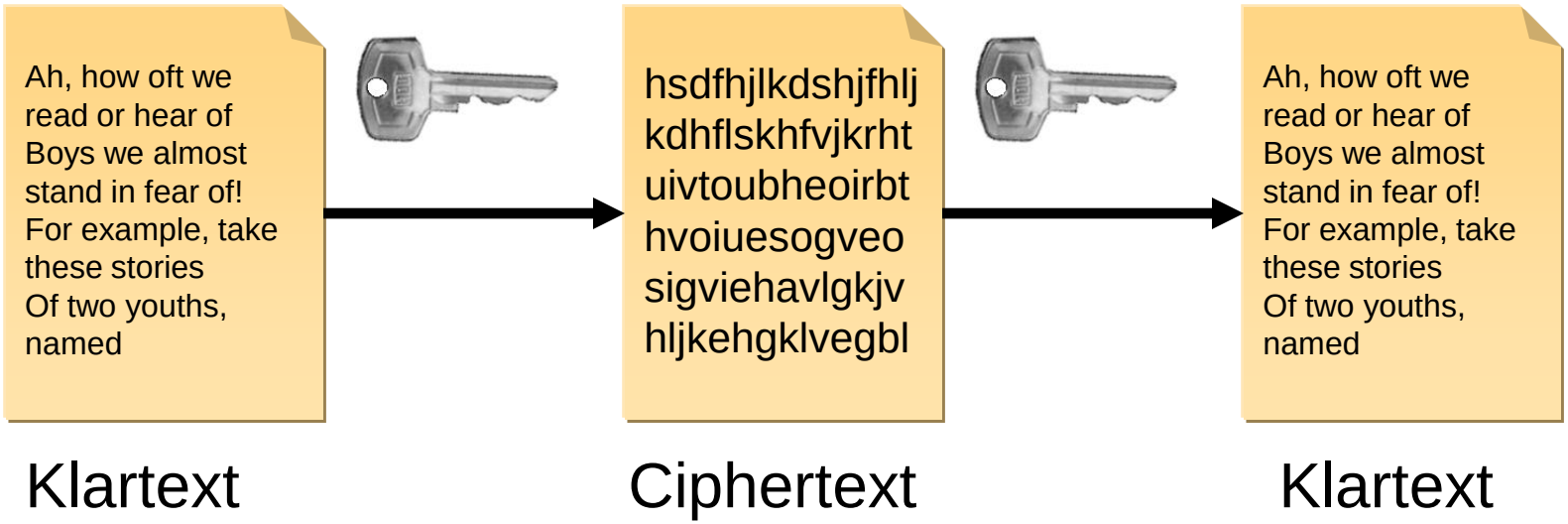
```
MAC=hash(Schlüssel+X)
```

Hashed Message Authentication Code (HMAC)

- Mac der auf kryptographischer Hash-Funktion basiert
- Combination e.g. random string, time and password is hashed and used as key
  - Key changes every time
- Siehe RFC 2104

Symmetrische Verschlüsselung

Symmetrische Schlüssel



Blowfish

- Symmetrischer Block Chiper Algorithmus
- 1993 von Bruce Schneier
- Weit verbreitet, wird aber immer mehr von Twofish oder AES abgelöst
- Public Domain
- Schlüssellänge 32-448 Bits
- Einer der schnellsten Algorithmen bis auf den Wechsel von Schlüsseln
  - PC -> Geeignet für Passwörter
- Großer Footprint von ca. 4KByte
- Sicher bei kurzen Klartexten

Twofish

- Von Bruce Schneier und Co
- War einer der fünf Top-Kandidaten für AES
- Schlüssellänge: 128-256 Bit
- Effizient
- Sicher (Stand 2007)
- Frei

International Data Encryption Algorithmus

- Patentiert in USA, Europa, ...
- Frei für nicht kommerzielle Nutzung
- Wurde 1990 von der ETH Zürich und Ascom entwickelt
- Schlüssellänge: 128 Bit

GOST

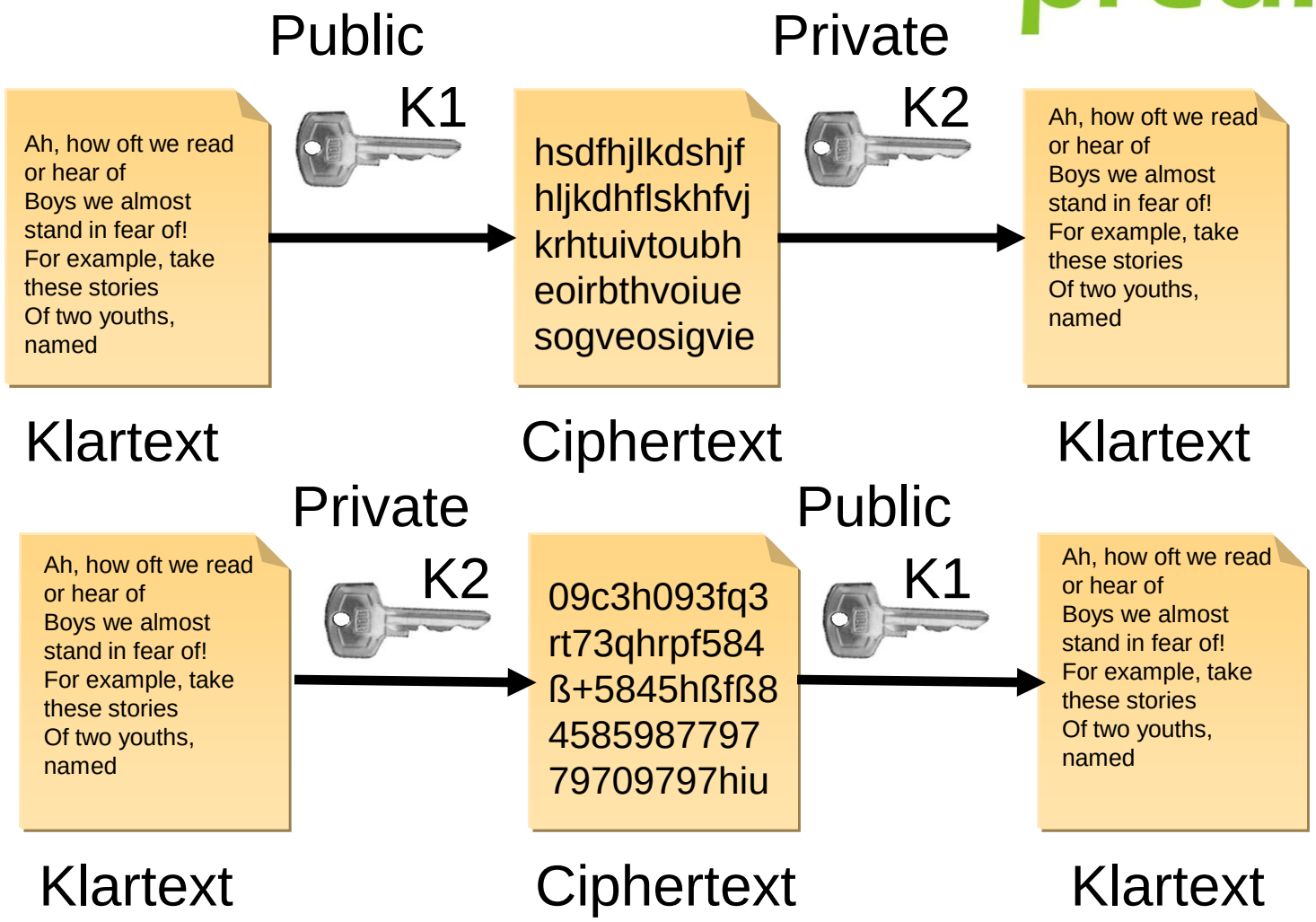
- Sowjetischer Regierungsstandard
- 1970 entwickelt
- Symmetrischer Block Cipher Algorithmus
- Gegenstück zum DES

Advanced Encryption Standard AES

- Aka Rijndael (von Rijmen u. Daemen)
- US Regierungsstandard für TOPSecretLevel
- Nachfolger von DES
- Symmetrisch
- Weit verbreitet
- Schnell und leicht zu implementieren
  - Für Software und Hardware
- Wenig Speicherverbrauch
- Schlüsselgröße: 128, 192 oder 256 Bits

Asymmetrische Verschlüsselung

Asymmetrische Schlüssel



Digital Signature Algorithm (DSA)

- US Regierungsstandard
- Erfunden von David W. Kravitz einem ehemaligen NSA Mitarbeiter
- Royalty free
- Basiert auf Primzahlen
- Arbeitet mit Public/Private Schlüsselpaar

RSA Algorithmus für Public-Key Verschlüsselung

- Geeignet für das Signieren und Verschlüsseln
- 1977 veröffentlicht im MIT von
  - Ron Rivest
  - Adi Shamir
  - Leonard Adleman
- Patent ist am 21. Sept. 2002 ausgelaufen
- Padding ist für kurze Klartexte notwendig
- Verschlüsseln und Entschlüsseln geht „zügig“

RSA Sicherheit des Algorithmus

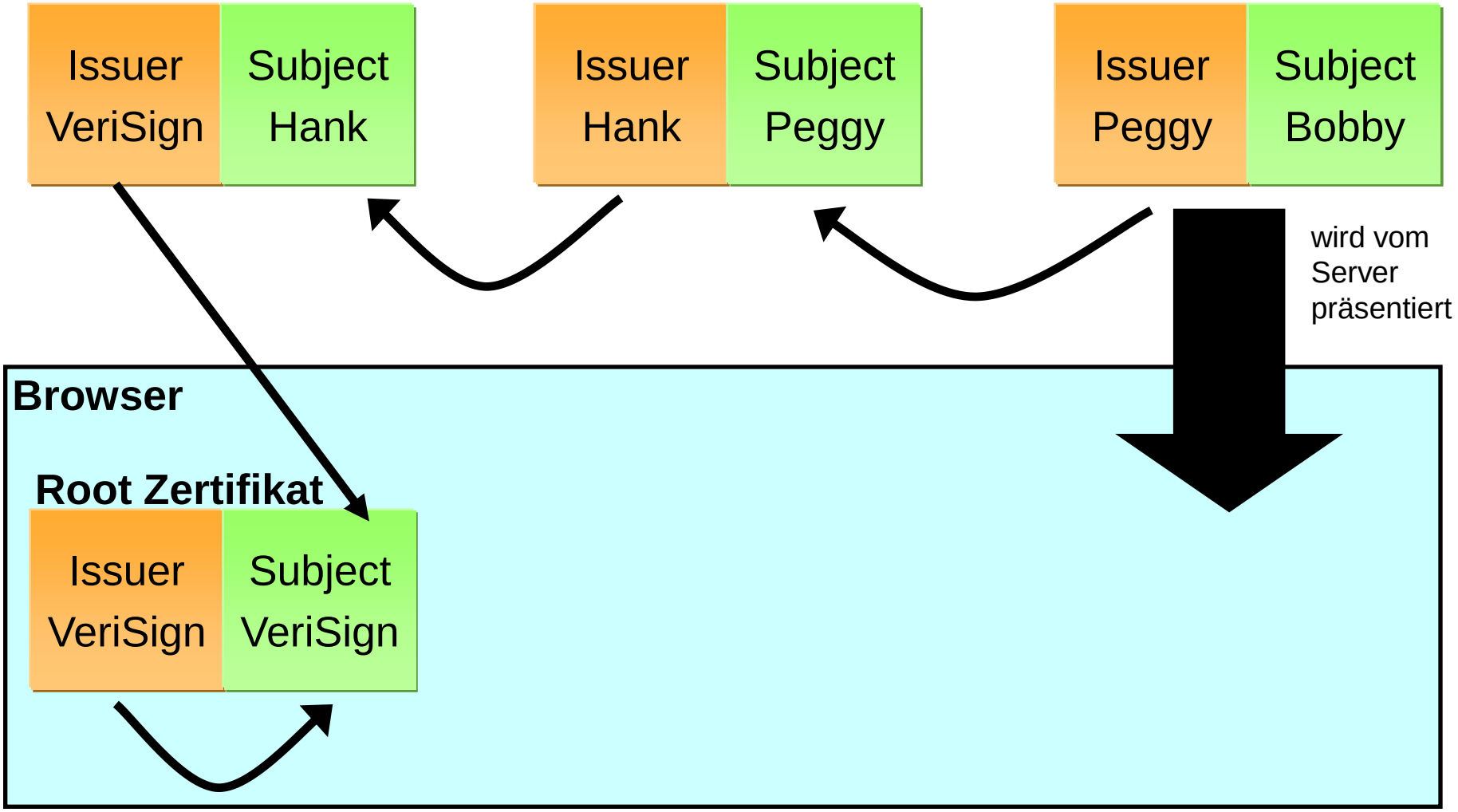
- RSA nutzt Primfaktorzerlegung
- 2005 war die längste zerlegte Zahl 663 Bits lang
- RSA Keys sind 1024-4096 Bits lang
- Es wird erwartet, daß 1024 Bit Keys bald gebrochen werden
- 2048 bzw. 4096 gelten als sehr sicher
- Ein Algorithmus von Peter Shor zeigt, daß ein Quanten Computer RSA in polynominaler Zeit entschlüsseln könnte.

Vergleich von Verschlüsselungsverfahren

	Asymetrische	Symetrische
Algorithmus	DSA, RSA	DES, AES, Blowfish
Geschwindigkeit	langsam	schnell
Schlüssellänge	lang – 4096 Bits	kurz
Anzahl Schlüssel	2	1

Public Key Kryptographie

Kette des Vertrauens
Beispiel: Zugriff mit SSL auf einen Server



Public-Key Zertifikat

- Ist signiert vom „Issuer“
- Gibt den Public-Key und eventuell andere Werte eines „Subjekts“ an
- Lösung für das Schlüssel-Verteilungs-Problem

Public-Key Infrastruktur

Software, Hardware, Organisation, Richtlinien und/oder Prozesse für das Erzeugen, Verwalten, Verteilen und Stornieren von Public-Key Zertifikaten.

Funktionen einer Public-Key Infrastruktur

- Registrierung
- Zertifizierung
- Schlüsselpaar Recovery
- Schlüsselerzeugung
- Schlüssel Aktualisierung
- Revocation
- Benachrichtigungen über Zertifizierungen und Stornierungen

Bestandteile einer Public-Key Infrastruktur

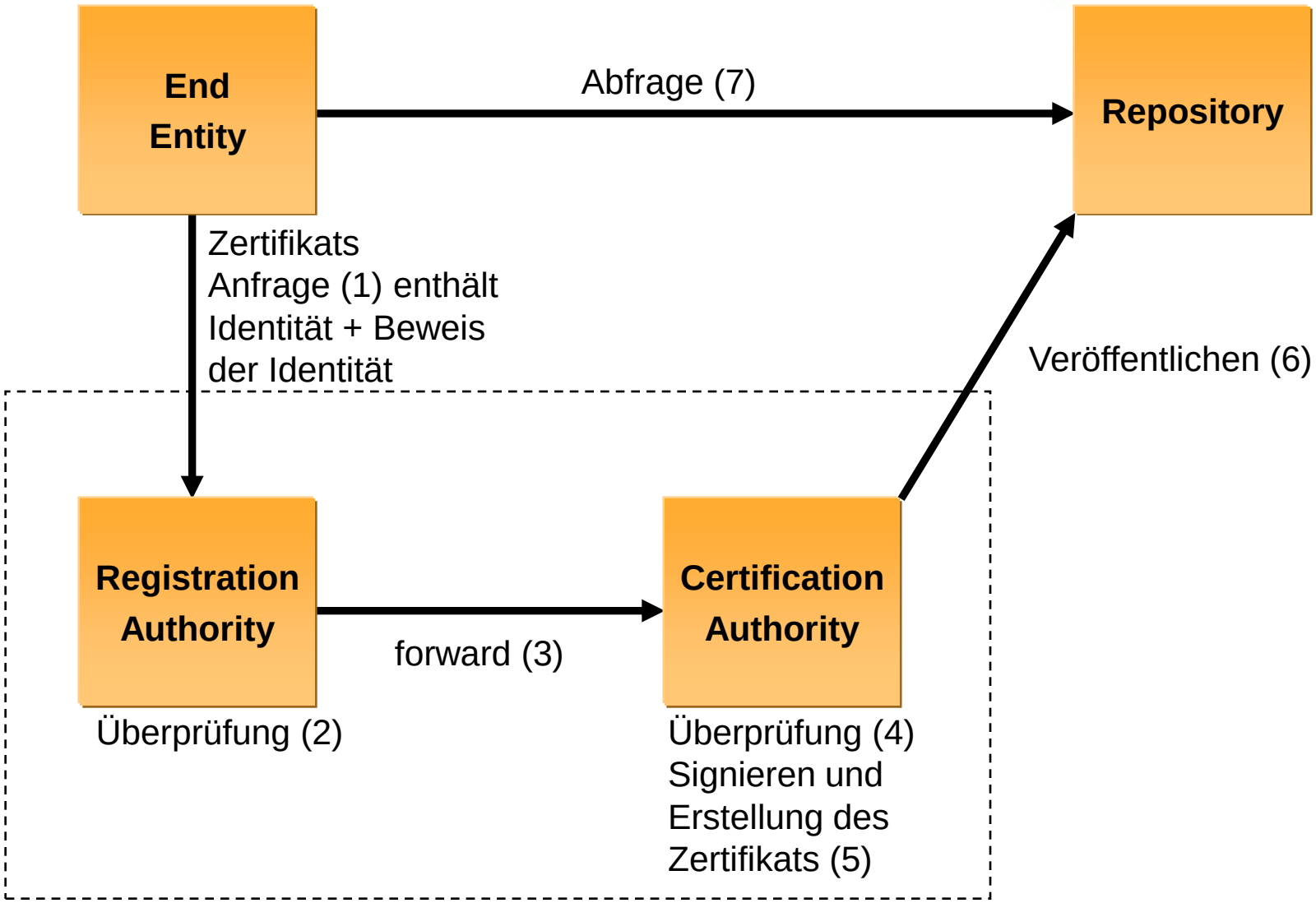
Teilnehmer	Tätigkeiten
Certification Authority	Ausstellen und Stornieren von Zertifikaten
Besitzer von Zertifikaten	Verschlüsseln und Signieren von Dokumenten
Client	Validieren von Signaturen bis zu vertrauenswürdigen CAs
Repository	Speichern von Zertifiakten und Revocation Listen

Certification Authority CA

- Vertrauenswürdiger Dritter
- Signiert Zertifikate für Dritte
- Root CAs
  - Entrust, Thawte, VeriSign
  - www.cacert.org (frei)
- Root Zertifikate in Java

  $JRE-HOME/lib/security/cacerts

Ablauf Zertifkat anfordern



Keystore

- Speicher für Schlüssel und Zertifikate
  - Eigene public/private Key Paare
  - Fremde public Keys
- Einträge
  - Schlüssel (Id, Private Key)
  - Trusted Certificates (Id, Public Key)
- Zugriff über
  - Java API
  - Keytool
- z.B. PKCS12 oder Sun JKS

Truststore

- Keystore, der Zertifikate von vertrauenswürdigen Entitäten enthält
- Wird ein Zertifikat importiert, so kommt dies den Vertrauen der jeweiligen Entität gleich

Key Agreement

- 2 Parteien etablieren gemeinsam Schlüssel
- Key Agreement Algorithmus
  - Diffie-Hellman (DH)
- Basiert auf beidseitig generierten Daten und privaten Schlüsseln

Cross-Zertifikate

- Zwei CAs tauschen sich aus, um ein Cross-Zertifikat zu erstellen

Proxy Zertifikate

- Werden von einer End Entität z.B. Benutzer ausgestellt
- Erweitern die Rechte einer anderen identität z.B. Rechner, Prozess, MP3Player

Der X.509 Standard

X.509

- ITU-T Standard für PKI
- Beschreibt Format für
  - Public-Key Zertifikate
  - Certification Path Validation Algorithm
  - Stornierung von Zertifikaten
- Der Begriff X.509 wird gewöhnlich benutzt für das PKIX Certificate Profile der IETF

Public-Key Infrastructure (PKIX)

- Working Group
- Arbeitsgebiete:
  - Profile für X.509v3 Zertifikate
  - Management Protokolle
  - Operational Protocols (LDAP, FTP, HTTP)
  - Certificate Policies
  - Time-Stamping, Data Certification/Validation

Standards und Protokolle, die X.509 Zertifikate verwenden

- TLS/SSL
- S/MIME
- IPSec
- SSH
- Smartcards
- LDAP
- Web Service Security
- XML Signature
- ...

Zertifikate in Windows

- Root Zertifikate sind bereits vorinstalliert
- Root Zertifikate können entfernt oder nachinstalliert werden
- Hash Codes der Root Zertifikate sind fest im Betriebssystem eingebaut
- IE oder Windows kann vorinstallierte Root Zertifikate nachladen, falls diese gelöscht wurden.
- Stornierungen von Zertifikaten werden u.U. nicht beachtet

X.500 Distinguished Names

- CommonName CN
- OrganisationUnit OU
- OrganisationName O
- LocalityName L
- StateName S
- Country C

- Beispiel:

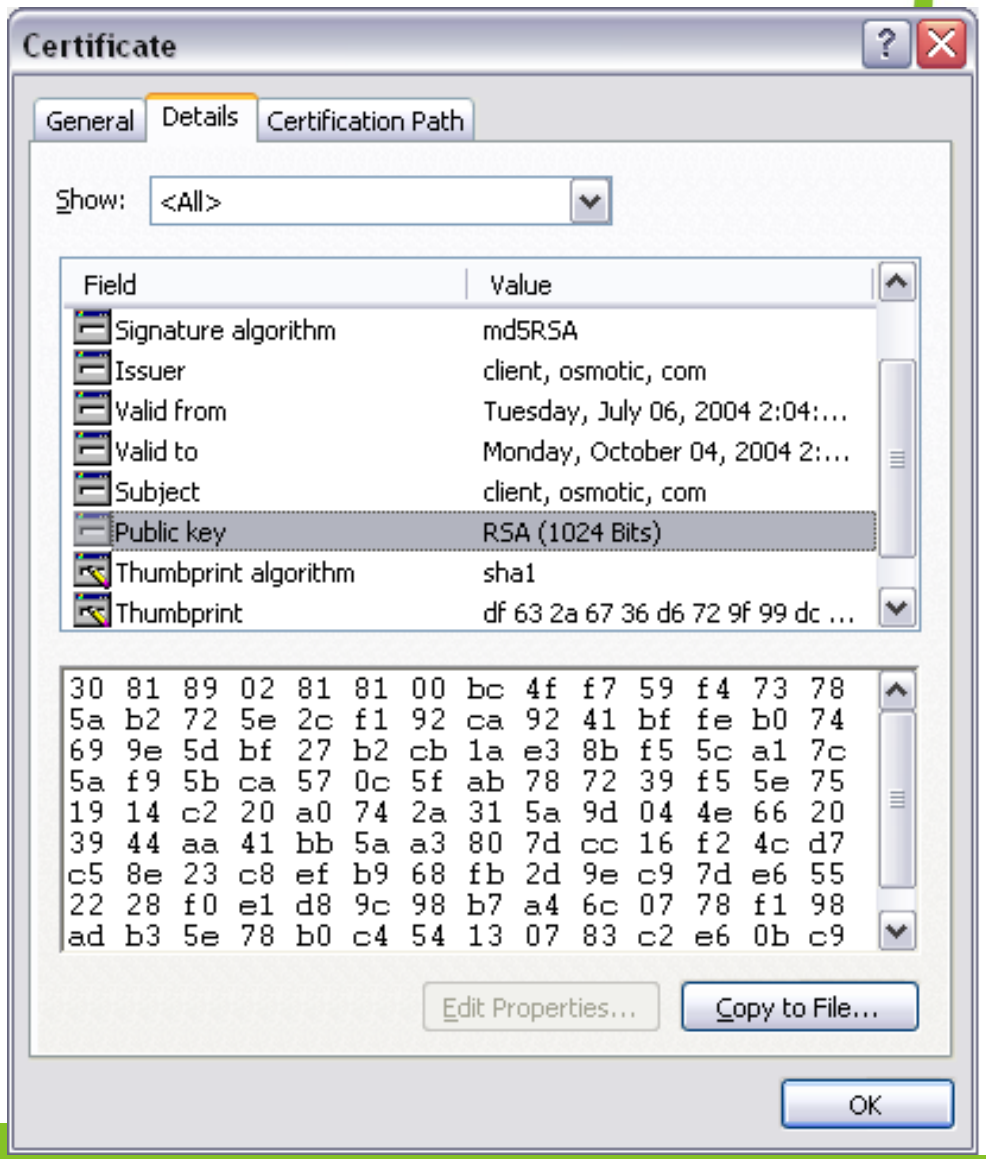
  CN=Hank Hill, OU=Sales, O=Striklen Propane, L=Arlen, S=Texas, C=US

Inhalt X.509 Zertifikat

- Version
  - Version des Standards V1, V2 oder V3
- Serial Number
- Signature Algorithm Identifier
- Issuer Name
  - X.500 Distinguished Name
- Validity Period
- Subject Name
- Subject Public Key Information
  - Wert
  - Algorithm Identifier
  - Parameter

Windows Zertifikat-Viewer

Zertifikat Detailansicht



PKCS Standards

Public Key Cryptography Standards PKCS

- Public Key Cryptography Standards von RSA Security
- Kein offizieller Standard, da von RSA Security kontrolliert

PKCS Standards

PKCS	Name	Bemerkung
#1	RSA Cryptography Standard	
#3	Diffie-Hellman Key Agreement Standard	
#5	Password-based Encryption Standard	
#6	Extended-Certificate Syntax Standard	Erweiterung zu X.509 v1 (jetzt obsolet von v3)
#7	Cryptographic Message Syntax Standard	Basis für S/Mime. Jetzt auf RFC 3852 basierend
#8	Private-Key Information Syntax Standard	
#9	Selected Attribute Types	
#10	Certification Request Standard	Format zu Zertifizierung von Public Keys von CAs anzufordern
#11	Cryptographics Token Interface (Cryptoki)	
#12	Personal Information Exchange Syntax	Dateiformat für Zertifikate
#15	Cryptographic Token Information Format Standard	

PKCS #12 Personal Information Exchange Syntax Standard

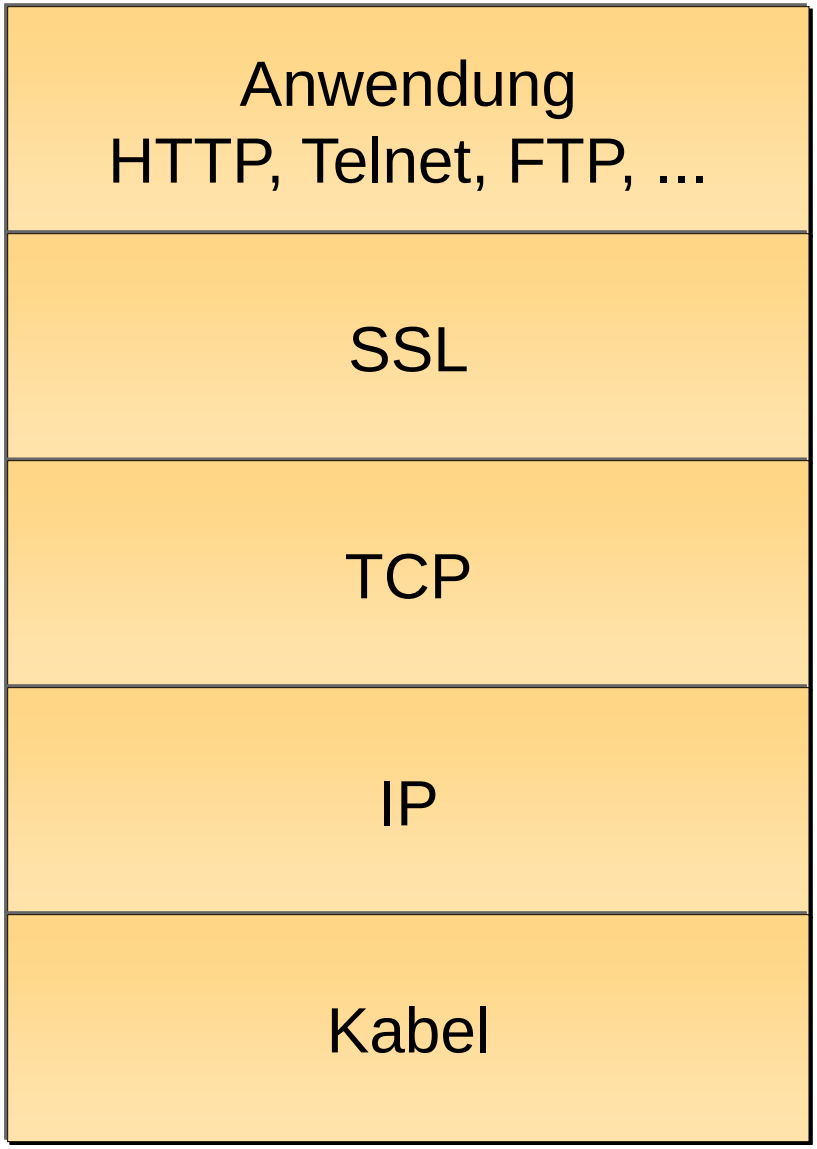
- RSA Security PKC Standard
- Dateiformat für Private Schlüssel mit dazugehörigen Public Key Zertifikat
- Geschützt durch Passwort basierten symmetrischen Schlüssel
- Nachfolger von PFX
- Verwendet in (Import/Export)
  - MS Internet Explorer
  - Firefox, Mozilla
  - ...

Secure Sockets Layer
SSL/TLS

Secure Sockets Layer (SSL)

- Internet Protokoll für Kryptographie im Web
- Größter Marktanteil
- Erweiterung für TCP/IP Sockets
- Ursprünglich von Netscape (1994)
- Basiert auf Public Key Kryptographie
- Bietet:
  - Two-Way Verschlüsselung
  - One-way oder Two-Way Authentifikation
  - Datenintegrität
  - Nicht: Non Reputiation
- Transport Layer Security TLS
  - Standardisierte SSL Variante
  - TLS 1.0 ist SSL 3.0 mit geringfügigen Änderungen

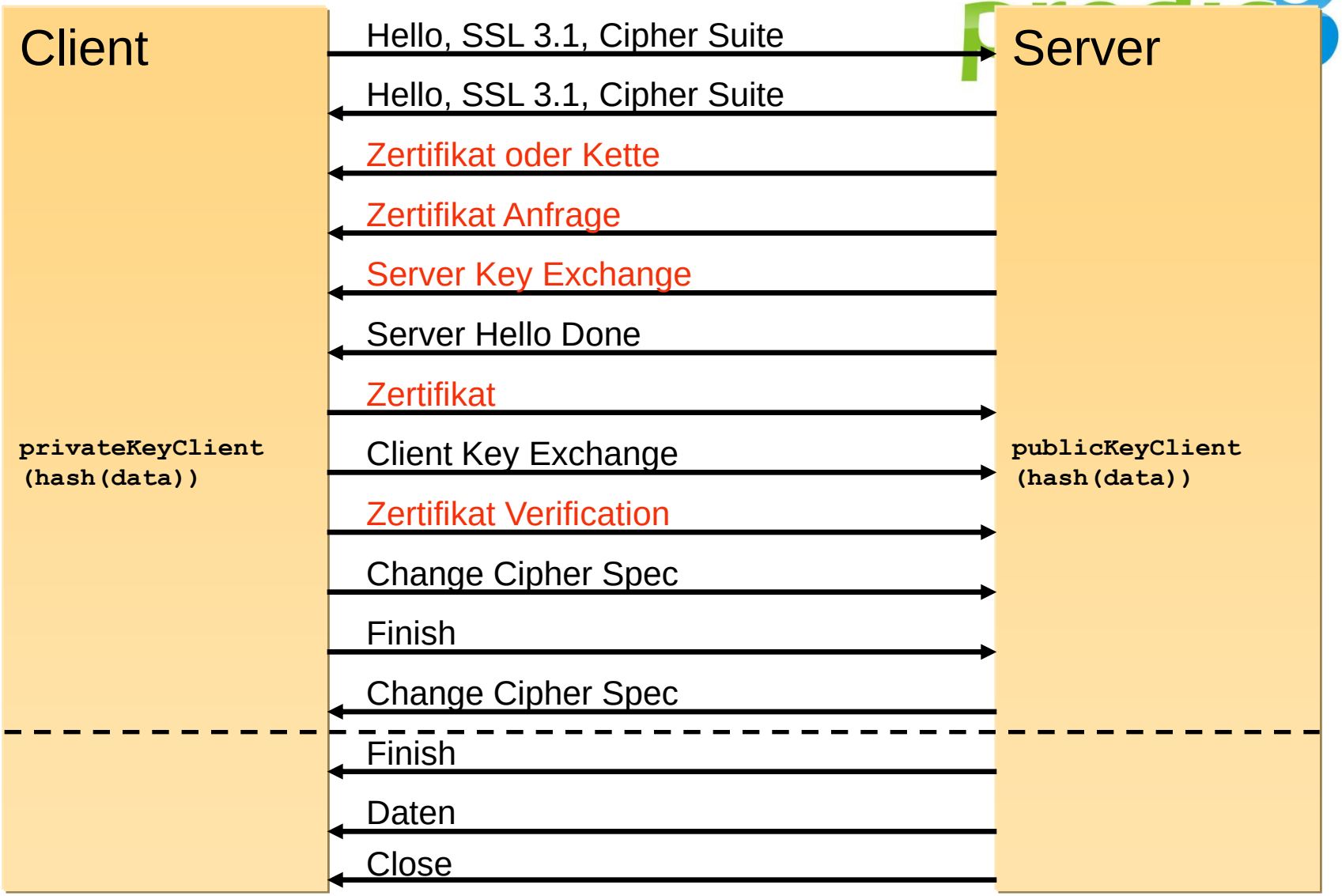
TCP/IP Stack mit SSL



SSL Handshake (vereinfacht)

- Einigung auf Chiper-Suite
  - Client präsentiert seine Chiper-Suite
  - Server wählt passende aus
- Authentifizierung (optional)
  - Client verlangt gewöhnlich Server Authentifizierung
  - Server verlangt selten Client Authentifizierung
- Austausch eines Sesses Keys

SSL Handshake



= optional

predic8

SSL/TLS mit Java und Apache Tomcat

Konfigurieren von SSL/TLS bei Client und Server

- Client
  - Keystore und Schlüssel erzeugen
  - System Properties setzen:
    javax.net.ssl.trustStore
    javax.net.ssl.keyStorePassword
  - Verändern der URL auf https und Port 443
- Server
  - Keystore und Schlüssel erstellen
  - Connector konfigurieren

SSL in Tomcat

- Client Authentication konfigurierbar
- Keystore Formate
  - Java Keystore JKS
  - PKCS12
- Je nach Konnektor Java oder OpenSSL(native connector) Implementierung

Client Authentification

```
<connector …
clientAuth="true|want|false"/>
```

Wert	Bedeutung
true	Client muss Zertifikat vorlegen
want	Client wird nach Zertifikat gefragt. Es gibt keinen Fehler falls Client kein Zertifikat vorweist.
false	Client wird nicht authentifiziert

SSL mit Jakarta HttpClient

- SSL u. TLS Unterstützung
  - Nutzt Java Secure Socket Extension (JSSE)
  - JSSE ist Bestandteil vom JRE ab Version 1.4
- Getunnelte SSL Verbindung über Proxy möglich
- Custom Socket Factories für:
  - Selfsigned Servers (EasySSL)
  - Client Authentication (AuthSSL)

Java Security

Security Komponenten in Java SE6

- JCA
- JCE
- JAAS
- Java Security Tools
- JSSE

Java Cryptography Architectures JCA

- Framework für kryptographische Komponenten
- Implementations unabhängig
- Algorithmen sind austauschbar

Unlimited Strength Java Cryptography

- Wegen Ausfuhrbeschränkungen enthält das JDK nur limitierte Kryptographie
- Sun bietet Download Bundle mit Jars, die signierte Policy-Dateien enthalten und unlimitierte Sicherheit ermöglichen

```
%JAVA_HOME%\lib\security\local.policy.jar
                         \US_export_policy.jar
```

Java Secure Socket Extension JSSE

- Framework und Implementation von SSL 3.0 und TLS 1.0
- Beinhaltet
  - Verschlüsselung
  - Server Authentifizierung
  - Nachrichtenintegrität
  - Client Authentifizierung
- Unterstützt
  - HTTP, FTP, telnet, TCP/IP
- Abstrahiert Algorithmen und „Handshaking“ Protokolle
- Seit JSE 1.4 Bestandteil von JDK
- Basiert auf JCA

SunJSSE Provider

- JSSE Implementierung von Sun
- Vorinstalliert in JCA
- SSL 3.0, TLS 1.0
- Unterstützt die verbreitesten Algorithmen
- X.509 basierter Key Manager und Trust Manager
- PKCS 12 Keystore

JSSE API

- Erweitert java.security und java.net um
  - Erweiterte Sockets
  - Trust- und Key-Manager
  - SSL Kontexte
  - Socket Factory Framework

Cryptographic Service Provider

- Stellt Algorithmen zur Verfügung
- Im JRE enthalten
- Stärkere Alternative:
  - http://www.bouncycastle.org/

Bouncy Castle Crypto Package

- Implementierung von Algorithmen
- MIT X Consortium Lizenz
- JCE Implementierung
- Algorithmen:
  - DES, IDEA, AES, Blowfish, RSA, SHA-X, Whirlpool, ...
- Beinhaltet Replacement Keystores
  - BKS, UBER, PKCS12

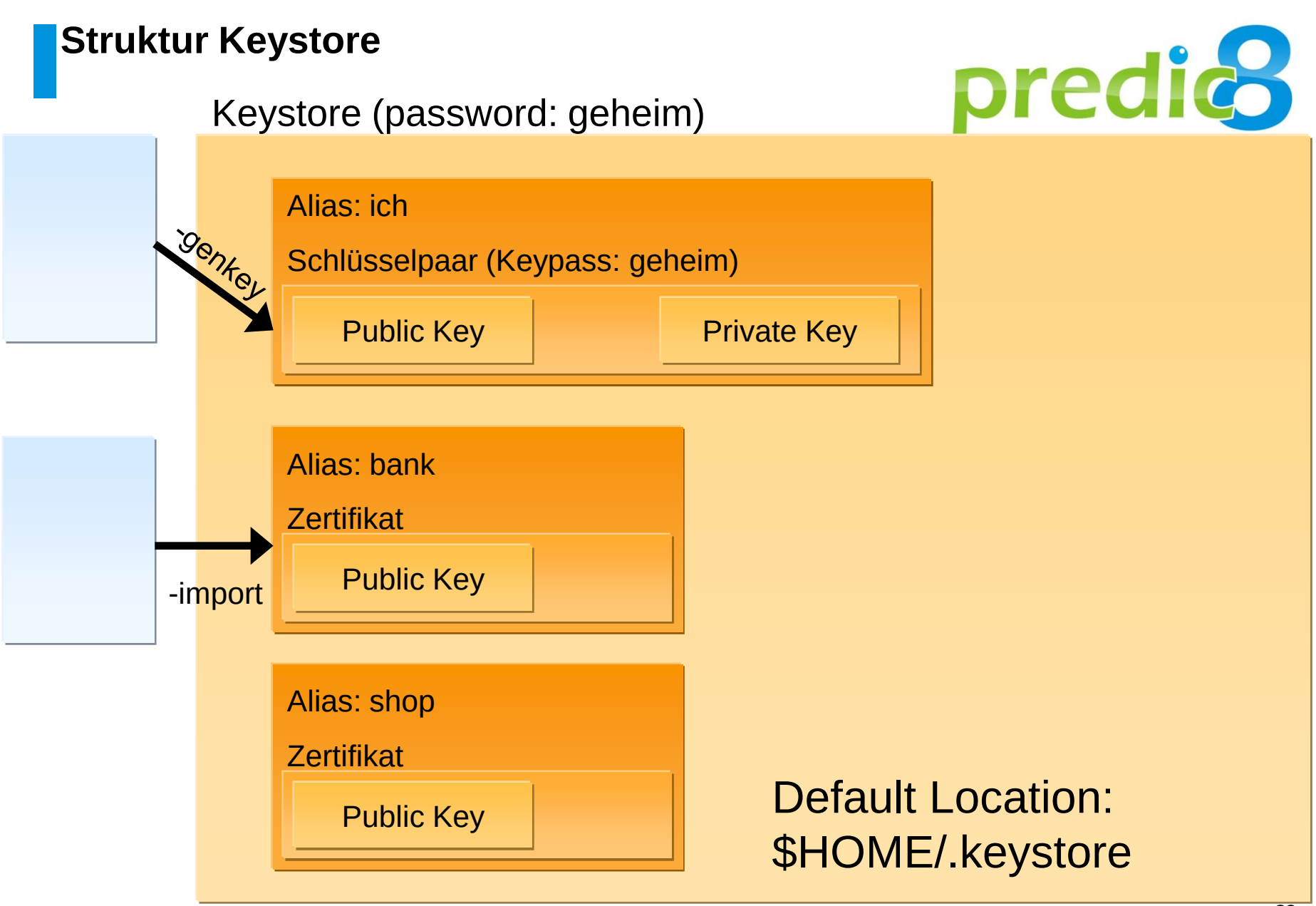
java.security Property Datei

- Ort: $JAVA_HOME/lib/security

```
security.provider.# =org.bouncycastle.jce.provider.BouncyCastleProvider
```

= Nummer abhängig von den installierten JCEs

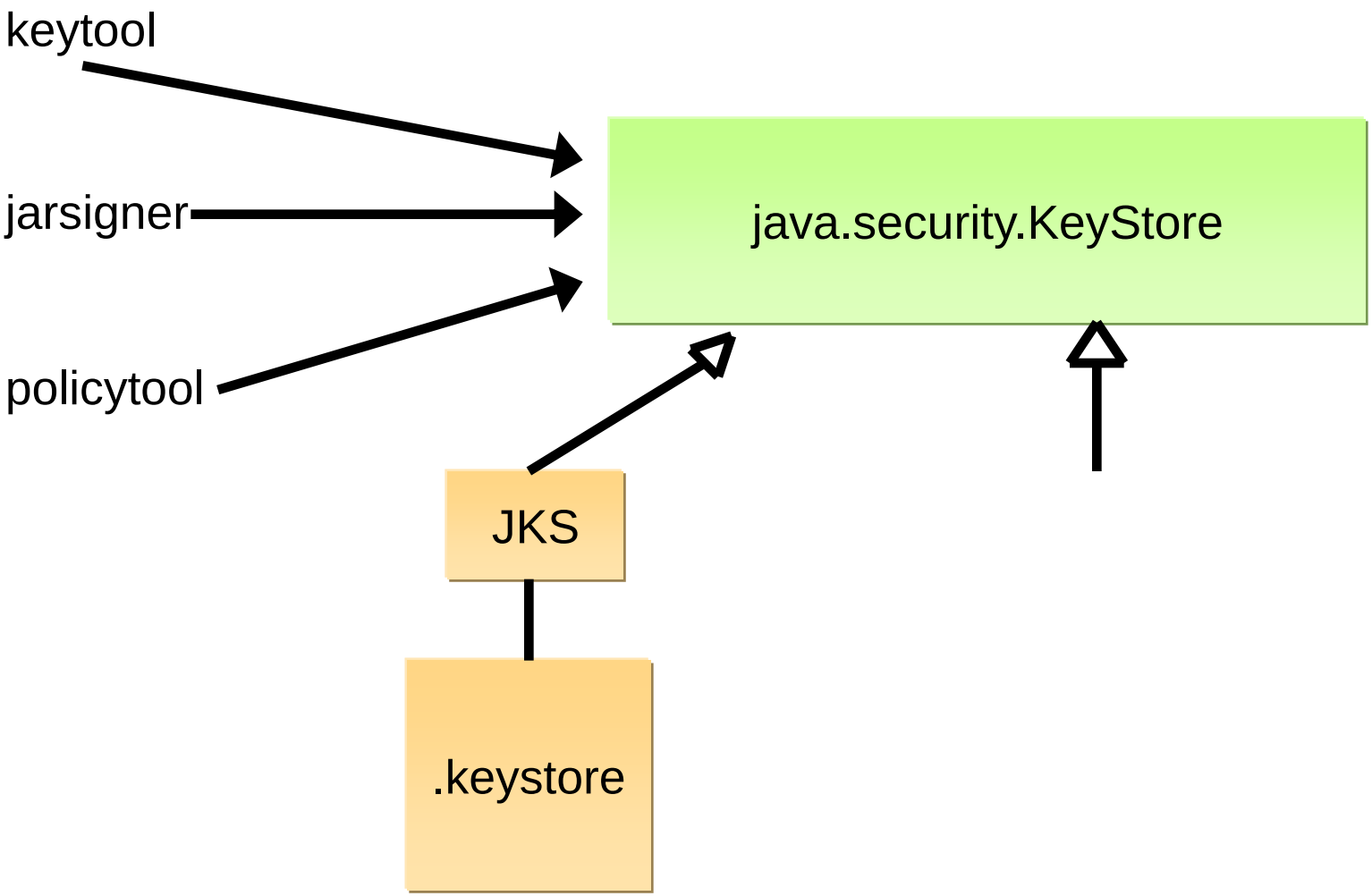
Struktur Keystore



Zugriff auf KeyStore mit Java

```
KeyStore ks = KeyStore.getInstance("JKS");

ks.load( new FileInputStream(file),
         storePass.toCharArray());

cert = (X509Certificate) ks.getCertificate(alias);

key = (PrivateKey) ks.getKey( keyAlias,
                              keyPass.toCharArray());
```

KeyStore



Java Keytool

- Bestandteil der Java SE
- Verwaltung eines Keystores
  - Eigene public/private Keys
  - Cache für Public keys
- Kommandozeile
- Default Algorithmus: DSA
- Default Schlüssel Größe: 1024

Erzeugen der Keystores

```
keytool -genkey -alias server -keyalg RSA
        -dname "CN=server, O=osmotic, C=com" -keypass simple
        -storepass simple -keystore server-store -storetype JKS

keytool -export -alias server -file server.cer
        -keystore server-store -storepass simple



keytool -import -noprompt -alias client -file client.cer
        -keystore server-store -storepass simple

keytool -import -noprompt -alias server -file server.cer
        -keystore client-store -storepass simple
```

jarsigner Tool

- Bestandteil von Java SE
- Generiert oder verifiziert Signaturen für JAR-Archive

JEE und Servlet basierte Sicherheit

Servlet Based Security

- Security can be integrated into a J2EE environment
- Tomcat can be used for security
  - Different realms can be used
    - **Memory**
    - **Database**
    - **LDAP**

Configuration

- Configure Tomcat realm
- Configure Web application to use servlet security
- Tell Axis to use servlet security
- Access usernames and roles inside Web Service implementations

Configuration in web.xml

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>AxisServlet</web-resource-name>
        <url-pattern>/services/BusTour</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>webservice</role-name>
    </auth-constraint>
</security-constraint>

<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>AxisServlet</realm-name>
</login-config>
<security-role>
    <description>the webservice role</description>
    <role-name>webservice</role-name>
</security-role>
```

Request

```
POST /axis/services/BusTour HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime,
multipart/related, text/*
User-Agent: Axis/1.2beta
Host: 127.0.0.1:2000
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 953
Authorization: Basic am9lOnNpbXBsZQ==
```

Response for Rejected Request

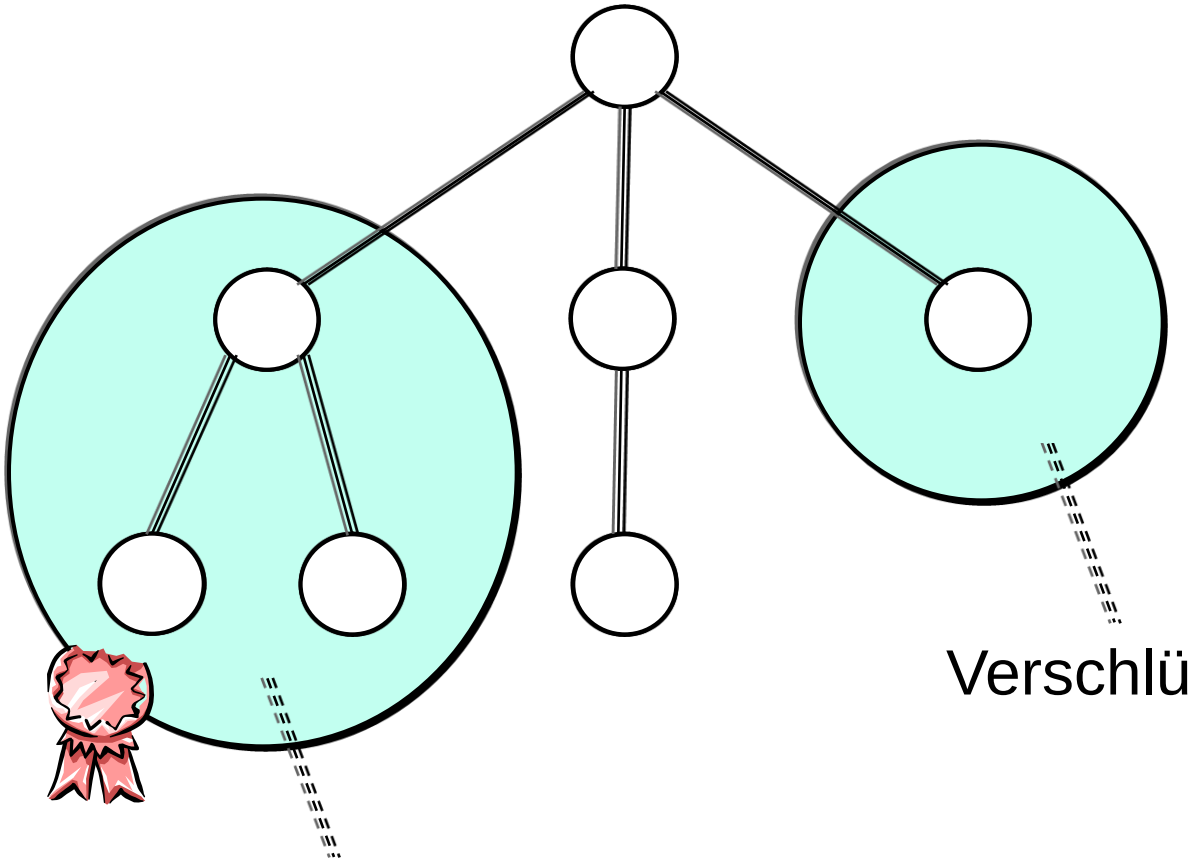
```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="AxisServlet"
Content-Type: text/html;charset=ISO-8859-1
Content-Language: en-US
Content-Length: 954
Date: Tue, 06 Jul 2004 13:55:21 GMT
Server: Apache-Coyote/1.1
Connection: close
```

Using Imperative Security

```
public void addTour(int id, String start,String destination, Date date) throws RemoteException{

    MessageContext ctx = MessageContext.getCurrentContext();

    SecurityProvider provider = (SecurityProvider)
        ctx.getProperty(MessageContext.SECURITY_PROVIDER);
    AuthenticatedUser au = provider.authenticate(ctx);
    if(au == null)
        throw new RemoteException("User is not authenticated");
    System.out.println("user: "+au.getName());
    if(!provider.userMatches(au, "conductor"))
        throw new RemoteException("Not Authorised");

}
```

XML Sicherheit

XML Security - Beispiele



XML Security Standards/ Spezifikationen

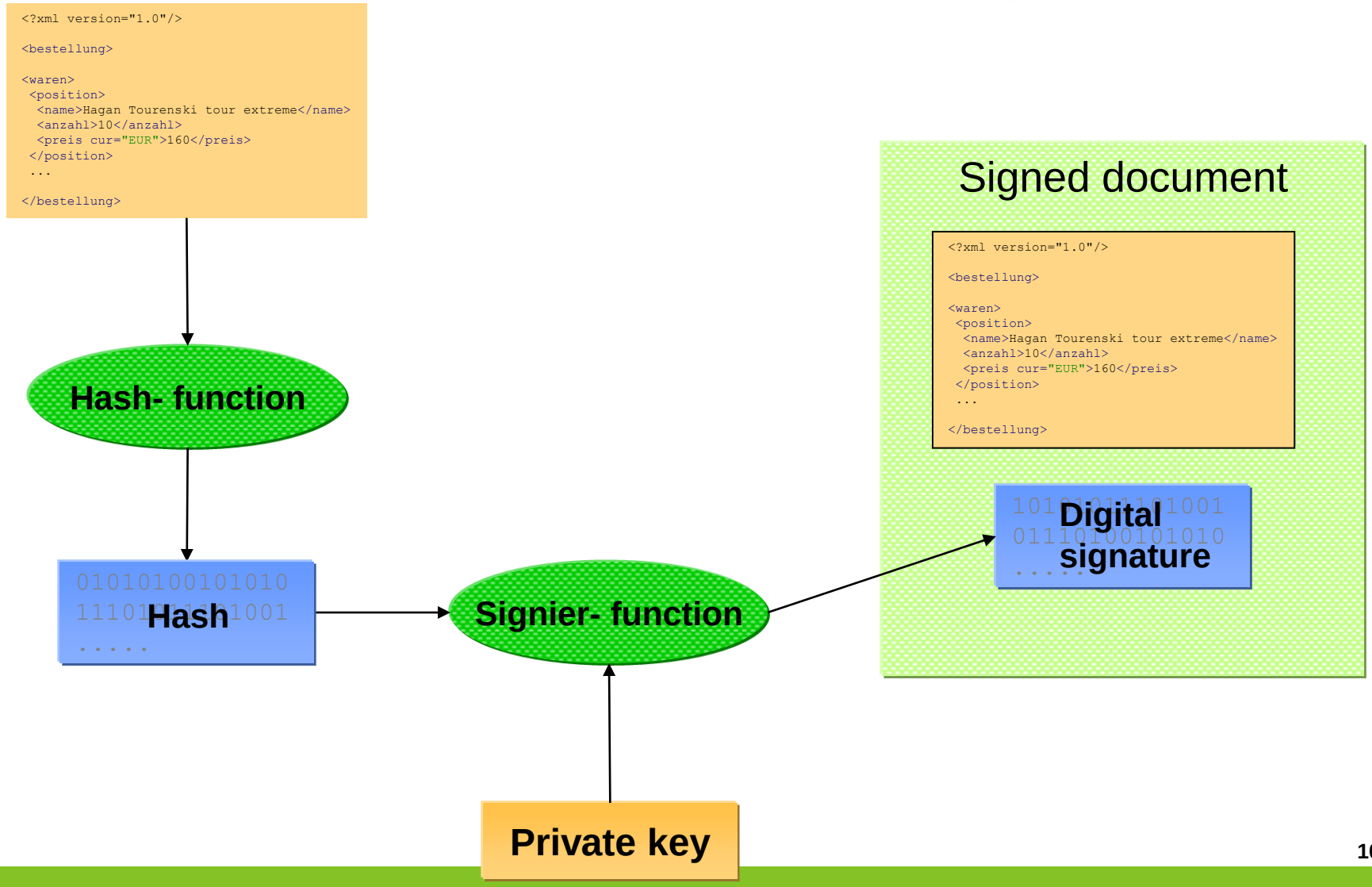
- XML Encryption
  - http://www.w3.org/Encryption/2001/
- XML Signature
  - http://www.w3c.org/Signature
- eXtensible Access Control Language (XACL)
  - http://www.trl.ibm.com/projects/xml/xacl/index_e.htm
- Security Assertion Markup Language (SAML)
  - http://www.oasis-open.org
- XML Key Management Specification (XKMS)
  - http://www.w3.org/TR/xkms/

XML Signature

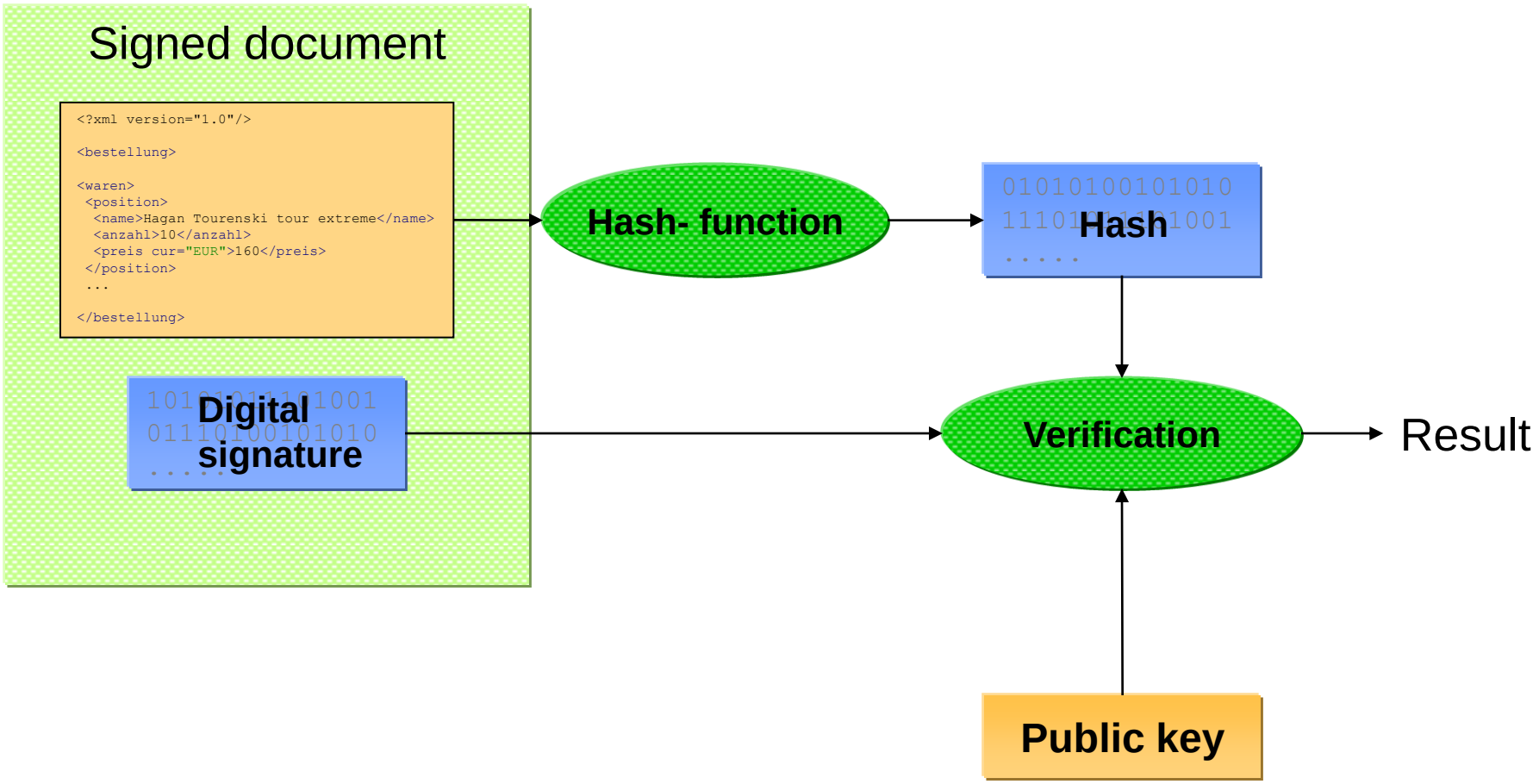
Digitale Signatur

- Unterschiedliche Legaldefinitionen von digitalen Signaturen
- Kryptographischer Wert
- Wird mit Daten versendet
- Mit einer Signatur kann sichergestellt werden:
  - Nachrist ist unverändert (Integrität)
  - Nachricht ist von angegebenen Absender

Generation of a Digital Signature



Digital Signature Validation



XML Signaturen

- W3C Standard seit Feb. 2002
- Bietet Integrität sowie Nachrichten und Signer Authentifizierung
- Für Daten innerhalb und außerhalb des Dokumentes

XML- „Gleichheit“

```
<root>
  <name type="1" source="site1">Herbert Hansen</name>
  <adress/>
</root>
```

CR/LF

```
<root>
  <name source="site1" type="1">Herbert Hansen</name>
  <!-- Hier steht die Adresse -->
  <adress></adress>
</root>
```

LF

XML Canonicalization (C14N)

- Bestimmt logische Gleichheit von Dokumenten
- Interessant für Verschlüsselung und Digitale Signaturen
- Standard vom W3C
  - http://www.w3.org/TR/xml-c14n

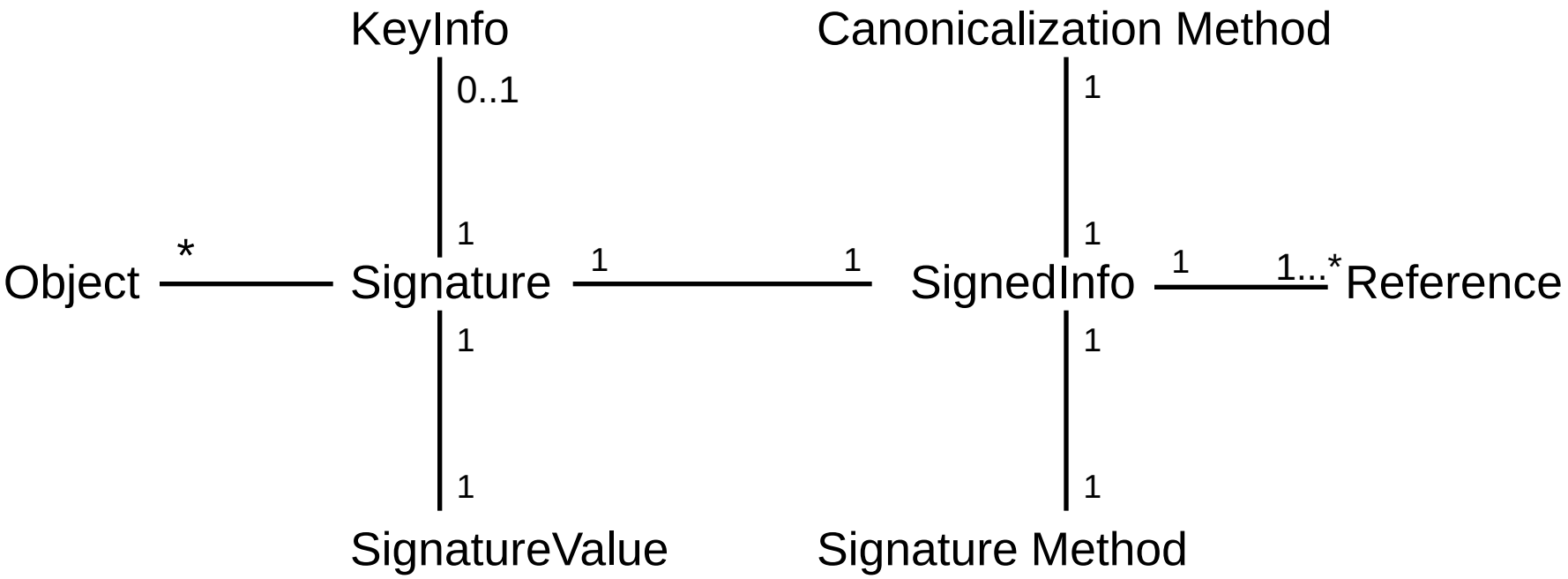
XML Signature Arten

- Detached
  - Unterschrift für externe Daten
- Enveloping
  - Unterschrift für Daten, die in einem `Object` Element der Signature enthalten sind.
- Enveloped
  - Unterschrift in zu signierenden Daten eingeschlossen

Enveloped Signature

- Signatur für Daten innerhalb eines Dokumentes
- Signatur ist Kind

Struktur Signature Dokument



Transformations

- Beispiele
  - C14N
  - Kompression
  - XSLT, XPath

Key Info

- Schlüssel zum Validieren der Signatur
  - Zertifikate
  - Schlüssel Namen
  - Algorithmen zur Schlüsselvereinbarung
- Optional

Erzeugen einer Signatur mit Apache XML Security

```
Init.init();
Document doc = createDocument();
XMLSignature sig = new XMLSignature(doc, null,
      XMLSignature.ALGO_ID_SIGNATURE_DSA);

doc.getDocumentElement().appendChild(sig.getElement());

Transforms tf= new Transforms(doc);

tf.addTransform(Transforms.TRANSFORM_ENVELOPED_SIGNATURE);
tf.addTransform(Transforms.TRANSFORM_C14N_WITH_COMMENTS);
sig.addDocument("", tf, Constants.ALGO_ID_DIGEST_SHA1);

KeyStore ks = getKeyStore();
sig.addKeyInfo(getCertificate(ks));
sig.sign(getPrivateKey(ks));
```

Erzeugte Signatur

```
<p8:order xmlns:p8="http://predic8.com/demo/">
  Dies ist die zu signierende Nachricht.
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm=„…/REC-xml-c14n-20010315„/>
      <ds:SignatureMethod Algorithm=„…#dsa-sha1„/>
      <ds:Reference URI="">
        <ds:Transforms>
          <ds:Transform Algorithm=„…#enveloped-signature„/>
          <ds:Transform Algorithm=„…#WithComments„/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm=„…#sha1„/>
        <ds:DigestValue>8PVzb/b4=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>N02wpngS4bvrXJ2XxGQ==</ds:SignatureValue>
    <ds:KeyInfo>
      <ds:X509Data>
        <ds:X509Certificate>MIYHKSDFSDFSDFDoZI</ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </ds:Signature>
</p8:order>
```

Validieren einer Signatur mit Apache XML Security

```
Init.init();

XMLSignature sig = getSignature(getDocument(file));
X509Certificate cert = sig.getKeyInfo().getX509Certificate();

if (sig.checkSignatureValue(cert)) {
  System.out.println("Signatur ist valide");
} else {
  System.out.println("Achtung! Signatur ist nicht valide");
}
```

XML Encryption

Verschlüsselte Teile eines Dokumentes

```
<Bestellung xmlns='http://www.oio.de/shop'>
  <Name>Herbert Hansen</Name>

  <Kreditkarte>
    <Nummer>4019 2445 0277 5567</Nummer>
    <Ablauf>12/03</Ablauf>
  </Kreditkarte>
</Bestellung>
```

```
<Bestellung xmlns='http://www.oio.de/shop'>
 <Name>Herbert Hansen</Name>
  <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
                 xmlns='http://www.w3.org/2001/04/xmlenc#'>
    <CipherData>
     <CipherValue>A23B45C56</CipherValue>
    </CipherData>
  </EncryptedData>
</Bestellung>
```

Verschlüsseln mit Apache XML Security

```
Init.init();

Document doc = loadDocument("data/creditcard.xml");

XMLCipher cipher = XMLCipher.getInstance(XMLCipher.TRIPLEDES);
Key key = KeyGenerator.getInstance("DESede").generateKey();
saveKey(key);

cipher.init(XMLCipher.ENCRYPT_MODE, key);
cipher.doFinal(doc, getCCNumber(doc), true);
```

Verschlüsselte Daten

```
<creditcard>
  <holder>Max Mustermann</holder>
  <ccnumber>
    <xenc:EncryptedData Type="…#Content">
      <xenc:EncryptionMethod  Algorithm="…#tripledes-cbc"/>
      <xenc:CipherData>
        <xenc:CipherValue>
          DiSBRcTxLbeJtTQ9B
       </xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </ccnumber>
</creditcard>
```

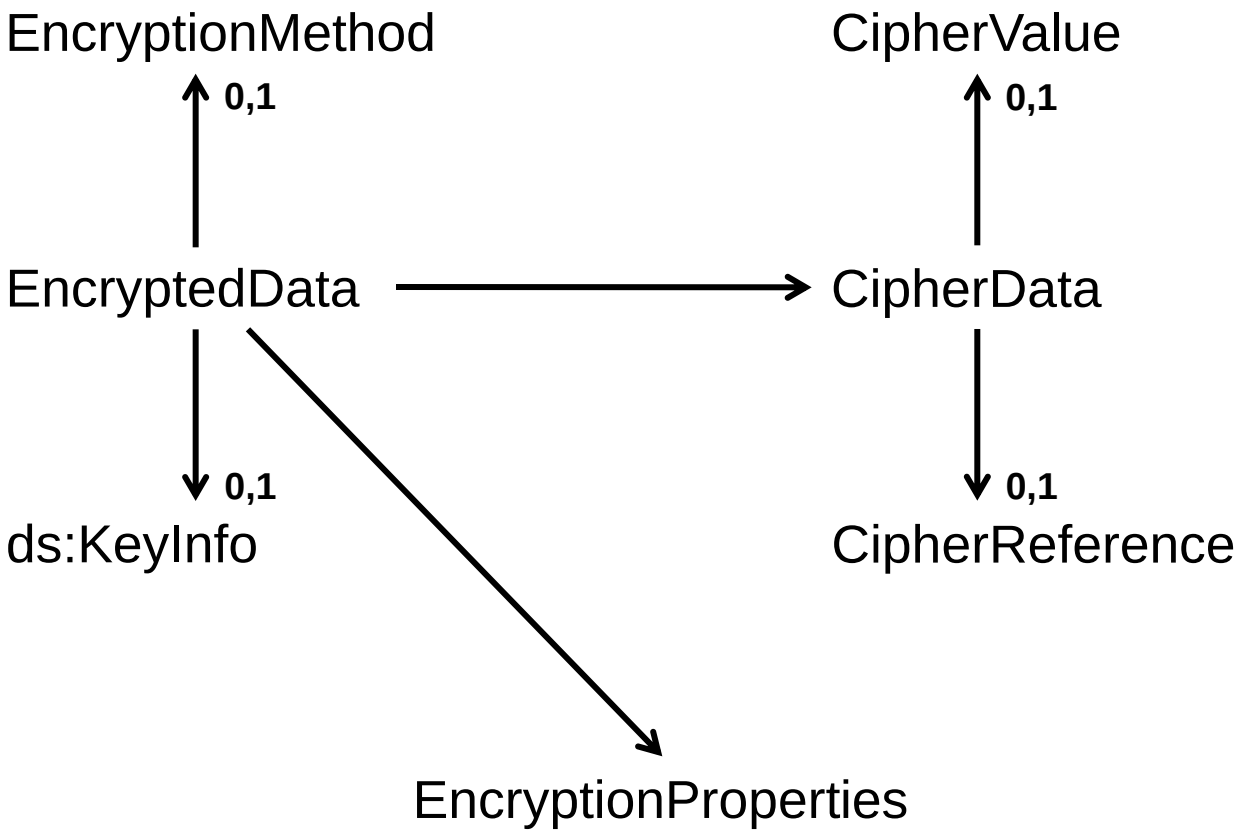
Entschlüsseln mit Apache XML Security

```
Init.init();

Document doc = loadDocument("encryptedInfo.xml");

XMLCipher cipher = XMLCipher.getInstance(XMLCipher.TRIPLEDES);
cipher.init(XMLCipher.DECRYPT_MODE, loadKey());
cipher.doFinal(doc, doc.getElementsByTagNameNS(
       EncryptionConstants.EncryptionSpecNS,
       EncryptionConstants._TAG_ENCRYPTEDDATA).item(0));
```

Verschlüsselte Daten

```
<xenc:EncryptedData Type="…#Element" Id="_5009">
  <xenc:EncryptionMethod Algorithm="…#aes128-cbc"/>
  <ds:KeyInfo xsi:type="keyInfo">
    <wsse:SecurityTokenReference>
      <wsse:Reference ValueType="…#EncryptedKey"
       URI="#_5002"/>
    </wsse:SecurityTokenReference>
  </ds:KeyInfo>
  <xenc:CipherData>
    <xenc:CipherValue>dbOtZQYDYlNUpQ4</xenc:CipherValue>
  </xenc:CipherDat>
</xenc:EncryptedData>
```

EncryptionData



Verschlüsselter Schlüssel

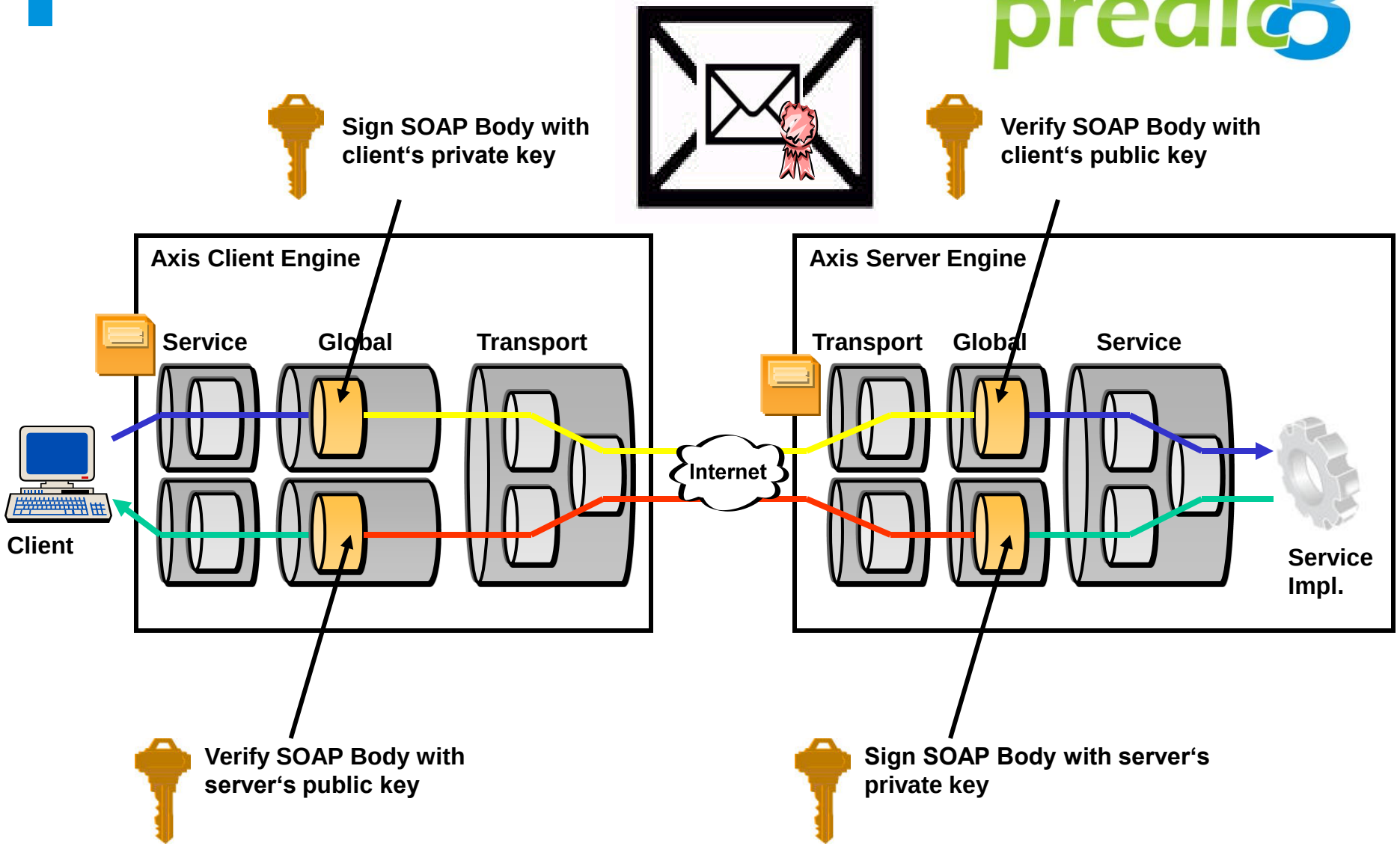
```
<xenc:EncryptedKey Id="_5002">
  <xenc:EncryptionMethod Algorithm="…#rsa-oaep-mgf1p"/>
  <ds:KeyInfo xsi:type="keyInfo">
    <wsse:SecurityTokenReference>
      <wsse:KeyIdentifier
       ValueType="…#X509SubjectKeyIdentifier"
       EncodingType="…#Base64Binary">
         dVEePzM6IWo=
      </wsse:KeyIdentifier>
    </wsse:SecurityTokenReference>
  </ds:KeyInfo>
  <xenc:CipherData>
    <xenc:CipherValue>bzc9dPf4n40C/Hc=</xenc:CipherValue>
  </xenc:CipherData>
</xenc:EncryptedKey>
```

Web Service Sicherheit

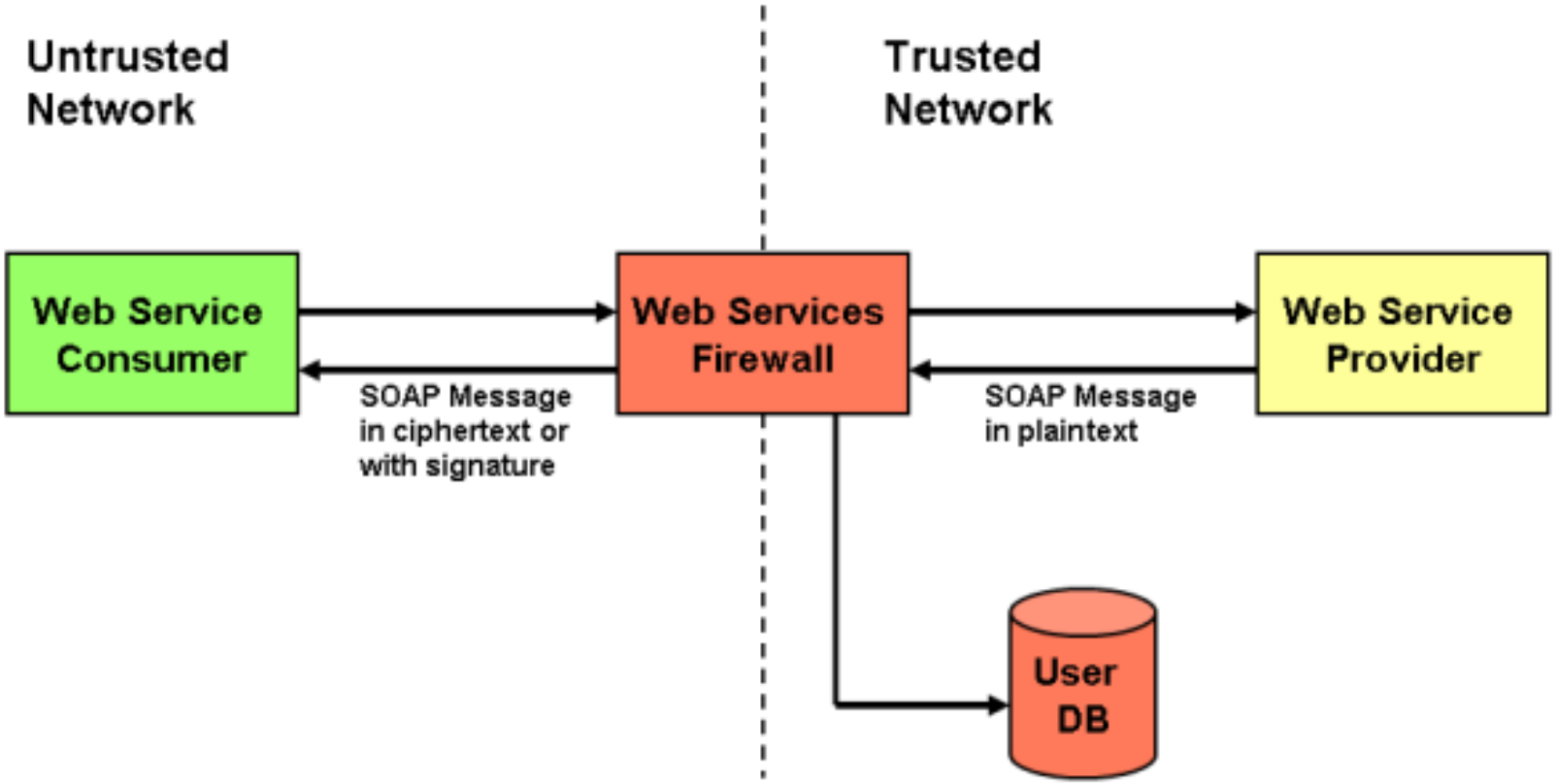
WS-I Basic Security Profile

- Version 1.1 vom 24.1.2010
- Focuses on
  - HTTP over TLS
  - OASIS Web Services Security 1.0
    - **Username Token Profile**
    - **X.509 Certificate Token Profile**
    - **Kerberos Token Profile**
    - **SAML Token Profile**
    - **Rights Expression Language REL**
  - Attachment Security

Transparent Signing and Encryption



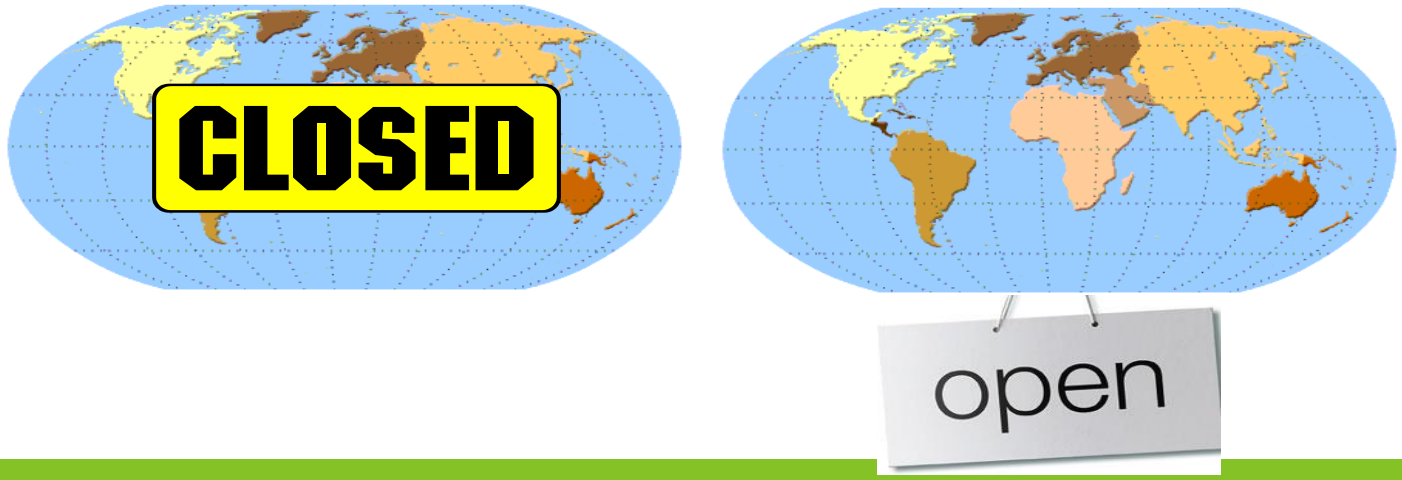
WS Firewall



How to Deal with Security

...[WS-Security]
will be useful for some,
but in the meantime the
existing SSL (Secure Sockets Layer)
security standard "solves 80 percent"
of security needs for Web services
...

Anne Thomas Manes, The Burton Group Corp



Der Web Services Security Standard

Aka WS-Security

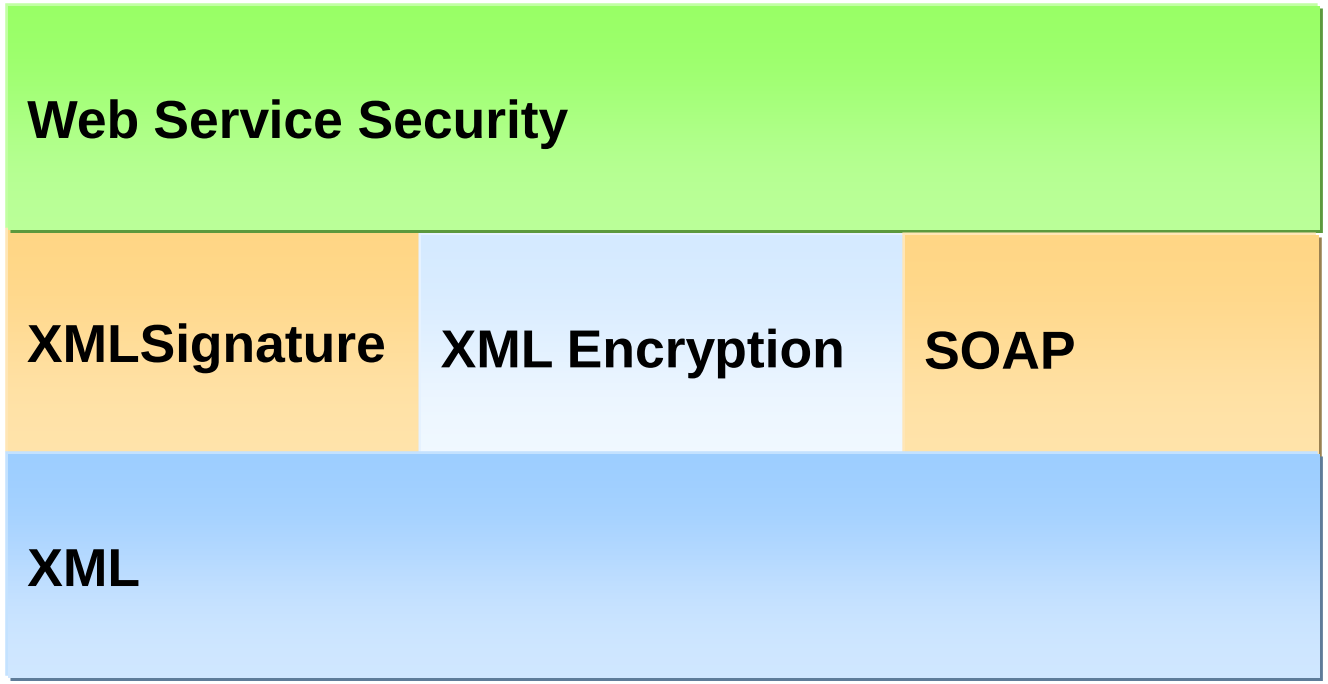
Web Services Security

- Sicherheit auf Nachrichtenbasis
- Version 1.1, Errata vom 1. Nov. 2006
- Erweiterungen zum SOAP Messaging für
  - Integrität
  - Vertraulichkeit
- Unterstützung für beliebige Token
- Beschreibt Mechanismen für das Erstellen von „Sicherheits Protokollen“

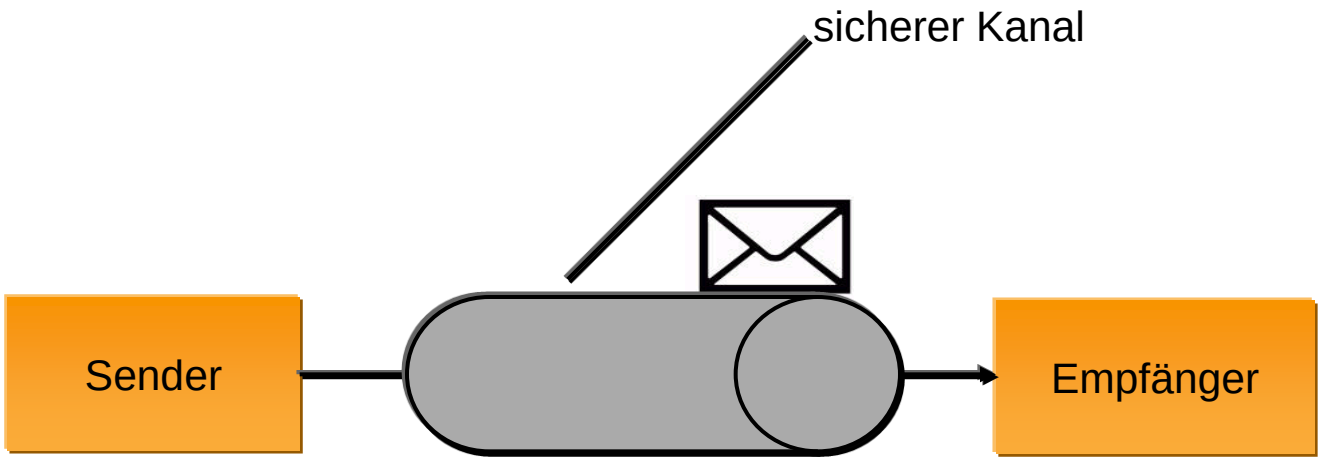
Nicht-Ziele für WSS

- Etablieren eines Security Contexts
- Schlüsselerzeugung und –Verteilung
- Bekanntmachen von Security Policies
- Non-Repudiation

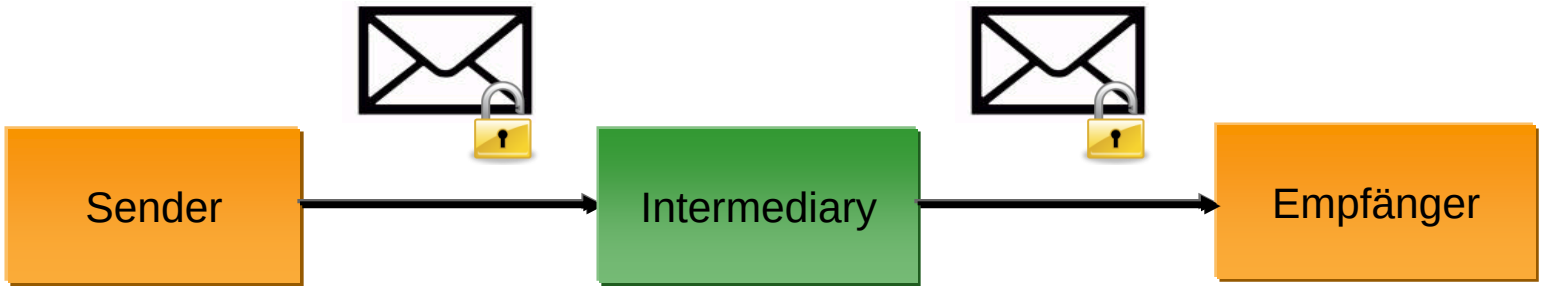
Web Services Security



Point-to-Point Transportlevel Security



End-to-End Nachrichtenbasierte Sicherheit



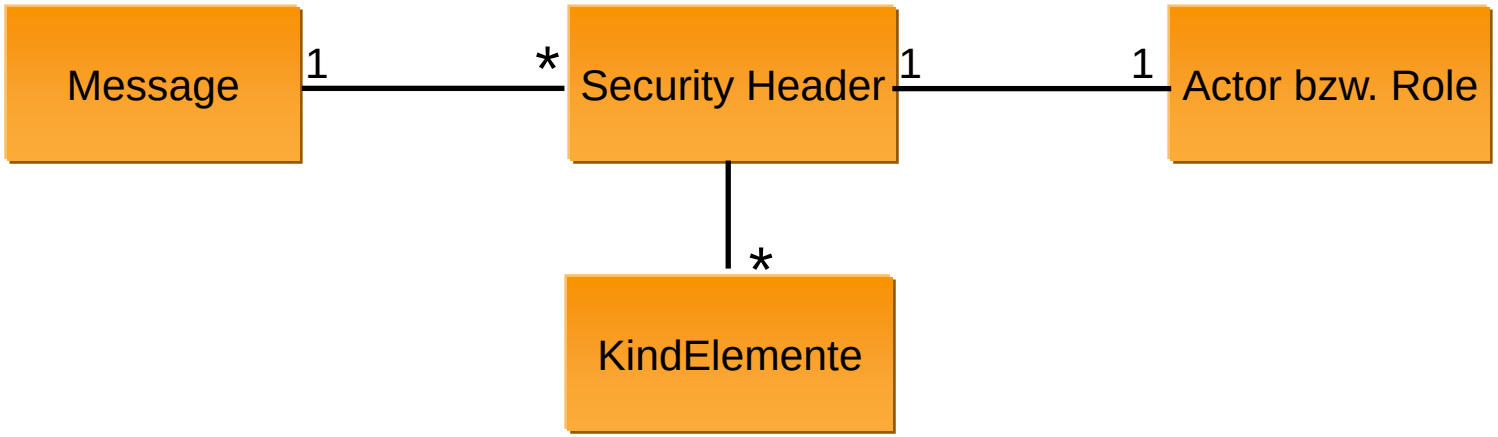
Verschlüsselte Nachricht

```
<S:Header>
  <wsse:Security S:mustUnderstand="1">
    <xenc:EncryptedKey Id=„symmetricKey">
      <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p"/>
      <ds:KeyInfo xsi:type="keyInfo">
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier ValueType="...#X509SubjectKeyIdentifier">server</wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>...</xenc:CipherData>
    </xenc:EncryptedKey>
    <xenc:ReferenceList>
      <xenc:DataReference URI="#payload"></xenc:DataReference>
    </xenc:ReferenceList>
  </wsse:Security>
</S:Header>
<S:Body wsu:Id="_5007">
  <xenc:EncryptedData Type="...#Element" Id="payload">
    <xenc:EncryptionMethod Algorithm="...#aes128-cbc"/>
    <ds:KeyInfo xsi:type="keyInfo">
      <wsse:SecurityTokenReference>
        <wsse:Reference ValueType="...#EncryptedKey" URI="#symmetricKey"/>
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
    <xenc:CipherData>...</xenc:CipherData>
  </xenc:EncryptedData>
</S:Body>
```

Security Header

- Beschreibt die Verschlüsselungs- und Signierungsschritte, die der Sender angewandt hat.
- Kind Elemente sind geordnet.

```
<S11:Header>
    ...
    <wsse:Security S11:actor="..." S11:mustUnderstand="...">
        ...
    </wsse:Security>
    ...
</S11:Header>
```



Username Token

- Spezifiziert Usernamen

```
<wsse:Security>
  <wsse:UsernameToken
    xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/xx/secext"
    xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/xx/utility">
    <wsse:Username>justus</wsse:Username>
    <wsse:Password Type="wsse:PasswordDigest">
    a3xLB2vVV=
    </wsse:Password>
    <wsse:Nonce>BLo234fg2=</wsse:Nonce>
    <wsu:Created> 2008-10-13T09:00:00Z </wsu:Created>
  </wsse:UsernameToken>
</wsse:Security>
```

Binary Tokens

- Binary Tokens müssen encodiert werden z.B.
  - X.509 Zertifakte
  - Kerberos Tickets
- ValueType: Beschreibt Token z.B. mit URI
- Encoding Type: Encodierung des Tokens #Base64Binary ist vordefiniert

```
<wsse:BinarySecurityToken wsu:Id=...
                          EncodingType=...
                          ValueType=.../>
```

ID Referenzen

- Verschlüsselte und signierte Elemente müssen referenzierbar sein
  - wsu:id
  - xml:id
  - lokale ID Attribute (XML Signature, XML Encryption)
- Eindeutig innerhalb eines Dokumentes

Referenzierung von Security Tokens

- Schlüssel oder Zertifikat in der Nachricht
- Seriennummer des Zertifikats
- Symbolischer Schlüsselname
- Subject Key Identifier

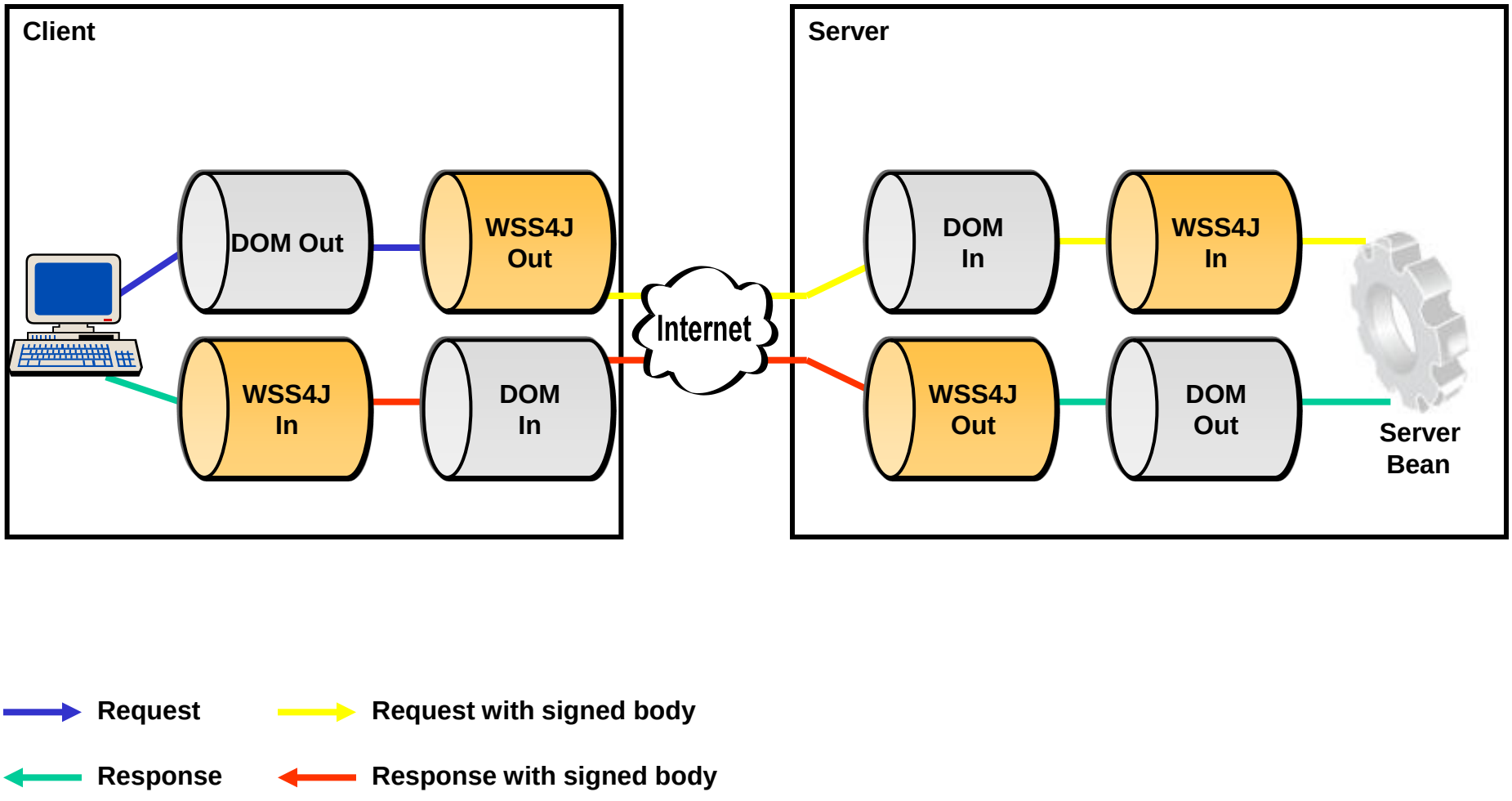
Subject Key Identifier SKI

- Können ab Version X.509v3 in Zertifikaten enthalten sein
- Aka Stellenschlüsselkennung in Windows (DE)

Security Token Reference

Ermöglicht Schlüssel innerhalb und außerhalb des Dokumentes zu referenzieren.

WSS Handler in XFire



Apache WSS4J

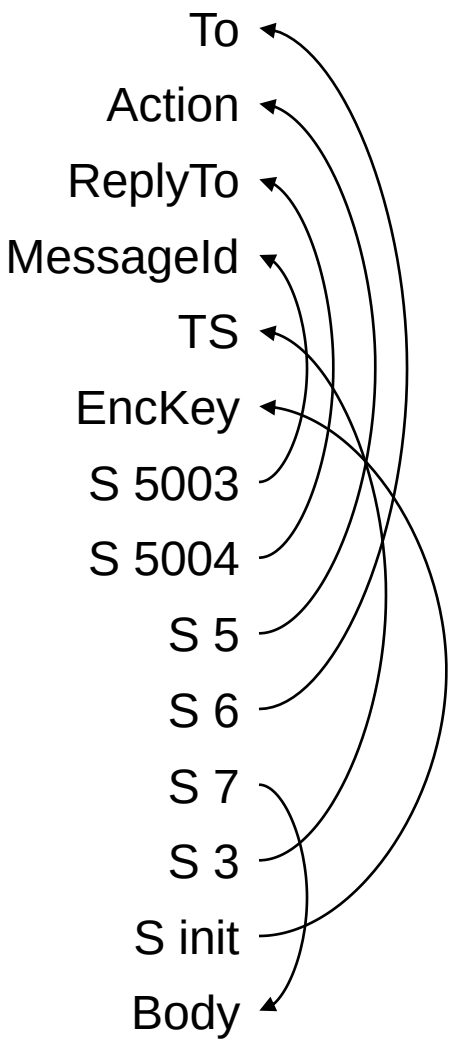
- Java Bibliothek für WSS
  - SOAP Message Security 1.0 200401
  - Username Token Profile V. 1.0
  - X.509 Token Profile V 1.0
- Wird verwendet von:
  - Axis
  - XFire

Outflow Config for signed and encrypted messages

```
<parameter name="OutflowSecurity">
  <action>
    <items>Timestamp Signature Encrypt</items>
    <user>alice</user>
    <passwordCallbackClass>org.apache.axis2.security.PWCallback</passwordCallbackClass>
    <signaturePropFile>interop.properties</signaturePropFile>
    <signatureKeyIdentifier>SKIKeyIdentifier</signatureKeyIdentifier>
    <encryptionKeyIdentifier>SKIKeyIdentifier</encryptionKeyIdentifier>
    <encryptionUser>bob</encryptionUser>
    <signatureParts>
      {Element}{http://schemas.xmlsoap.org/ws/2004/08/addressing}To
    </signatureParts>
    <optimizeParts>
      //xenc:EncryptedData/xenc:CipherData/xenc:CipherValue
    </optimizeParts>
  </action>
</parameter>
```

Quelle: Axis 2_0 - Rampart WS-Security module for Axis2

Client Request



Signatur eines SOAP Headers

```
<ds:Signature Id="_1">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
      <exc14n:InclusiveNamespaces PrefixList="wsse S"/>
    </ds:CanonicalizationMethod>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
    <ds:Reference URI="#_5003">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <exc14n:InclusiveNamespaces PrefixList="S"/>
        </ds:Transform>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>m0fi6JgZENlHslBgD0JVKQLvojI=</ds:DigestValue>
    </ds:Reference>
    <ds:Reference URI="#_5004">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <exc14n:InclusiveNamespaces PrefixList="S"/>
        </ds:Transform>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>5Ab1ebo4/FraGgck/A8iDx1J9+I=</ds:DigestValue>
    </ds:Reference>
    ...
  </ds:SignedInfo>
  <ds:SignatureValue>qJHk3xGvcG+8L4e0WuLPf6NKDfU=</ds:SignatureValue>
  <ds:KeyInfo>
    <wsse:SecurityTokenReference wsu:Id="uuid_18ca539b-9b6c-4484-8f63-c38fda32f2b9">
      <wsse:Reference ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#EncryptedKey"
URI="#_5002"/>
    </wsse:SecurityTokenReference>
  </ds:KeyInfo>
</ds:Signature>
```

WS-SecurityPolicy

WS-SecurityPolicy

- OASIS Standard (Version 1.2 vom 1.7.2007)
- Ergänzt WS-Policy
- Beschreibt Assertions für
  - WSS 1.0, 1.1
  - WS-Trust
  - WS-SecureConversation

Vordefinierte Security Bindings

- Transport Binding
- Symmetric Binding
- Asymmetric Binding

Security Binding Informationen

- Benötigt Token und deren Binding
- Vorgeschriebener Transport
- Benötigte Nachrichtenelemente z.B. Timestamp
- Inhalt des wsse:Security Headers
- Parameter z.B. Verwendete Algorithmen

Herausgeber eines Tokens

- Issuer
  - Über WSA Endpoint Reference
- Issuer Name

Inclusion of Token Assertions

- Never
- Once
- Always To Recipient
- Alsways To Initiator
- Always

Protection Assertion

- Integrity Assertions
  - Signal Ports
    - **Header, Body, Attachments**
  - SingedElements
    - **Beliebiges Element**
- Confidentiality Assertions
  - EncrytedPorts
  - EncryptedElements
  - ContentEncryptedElements
- Required Elements Assertion
  - RequiredParts
  - RequiredElement

Token Assertions

- Username Token
- Issued Token
  - Über WS-Trust bezogenes Token
- X509Token
- Verbose Token
- Spnego Context Token
- Security Context Token
- Secure Conversation Token
- SAML Token
- RelToken
- HTTPS Token
- Key Value

Security Header Layout

- Strict
  - In WSP 6.7.1 definiert
  - LAX, Reihenfolge muss nur WSS konform sein
  - LAX Timestamp First
    - **Wie LAX nur Timestamp zuerst**
  - Lax Timestamp Last

Security Binding Assertion

- Algorithm Suite
- Layout Assertion
- Transport Binding
- Symmetric Binding
- Asymmetric Binding

Policy für Vertraulichkeit und Integrität

```
<wsp:Policy
wsu:Id="StockQuoteBinding_GetStockQuote_Input_Policy">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:EncryptedParts>
        <sp:Body />
      </sp:EncryptedParts>
      <sp:SignedParts>
        <sp:Body />
        <sp:Header Name="To"
Namespace="http://www.w3.org/2005/08/addressing" />
        <sp:Header Name="From"
Namespace="http://www.w3.org/2005/08/addressing" />
</sp:SignedParts>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

SymmetricBinding Policy

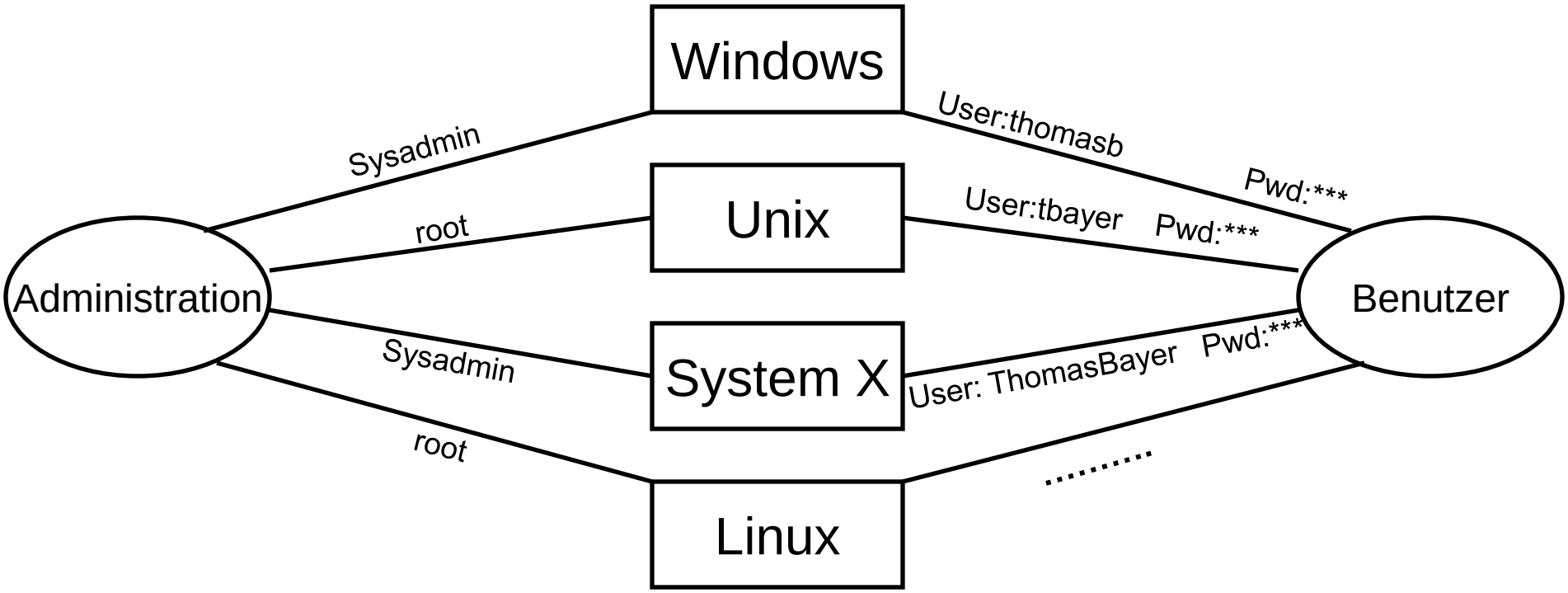
```
<sp:SymmetricBinding>
       <wsp:Policy>
         <sp:ProtectionToken>
           <wsp:Policy>
             <sp:X509Token sp:IncludeToken="…/IncludeToken/Never">
               <wsp:Policy>
                 <sp:WssX509V3Token10 />
               </wsp:Policy>
             </sp:X509Token>
           </wsp:Policy>
         </sp:ProtectionToken>
         <sp:Layout>
           <wsp:Policy>
             <sp:Strict />
           </wsp:Policy>
         </sp:Layout>
         <sp:IncludeTimestamp />
         <sp:OnlySignEntireHeadersAndBody />
         <sp:AlgorithmSuite>
           <wsp:Policy>
             <sp:Basic128 />
           </wsp:Policy>
         </sp:AlgorithmSuite>
       </wsp:Policy>
     </sp:SymmetricBinding>
```

SignedSupportingTokens Policy

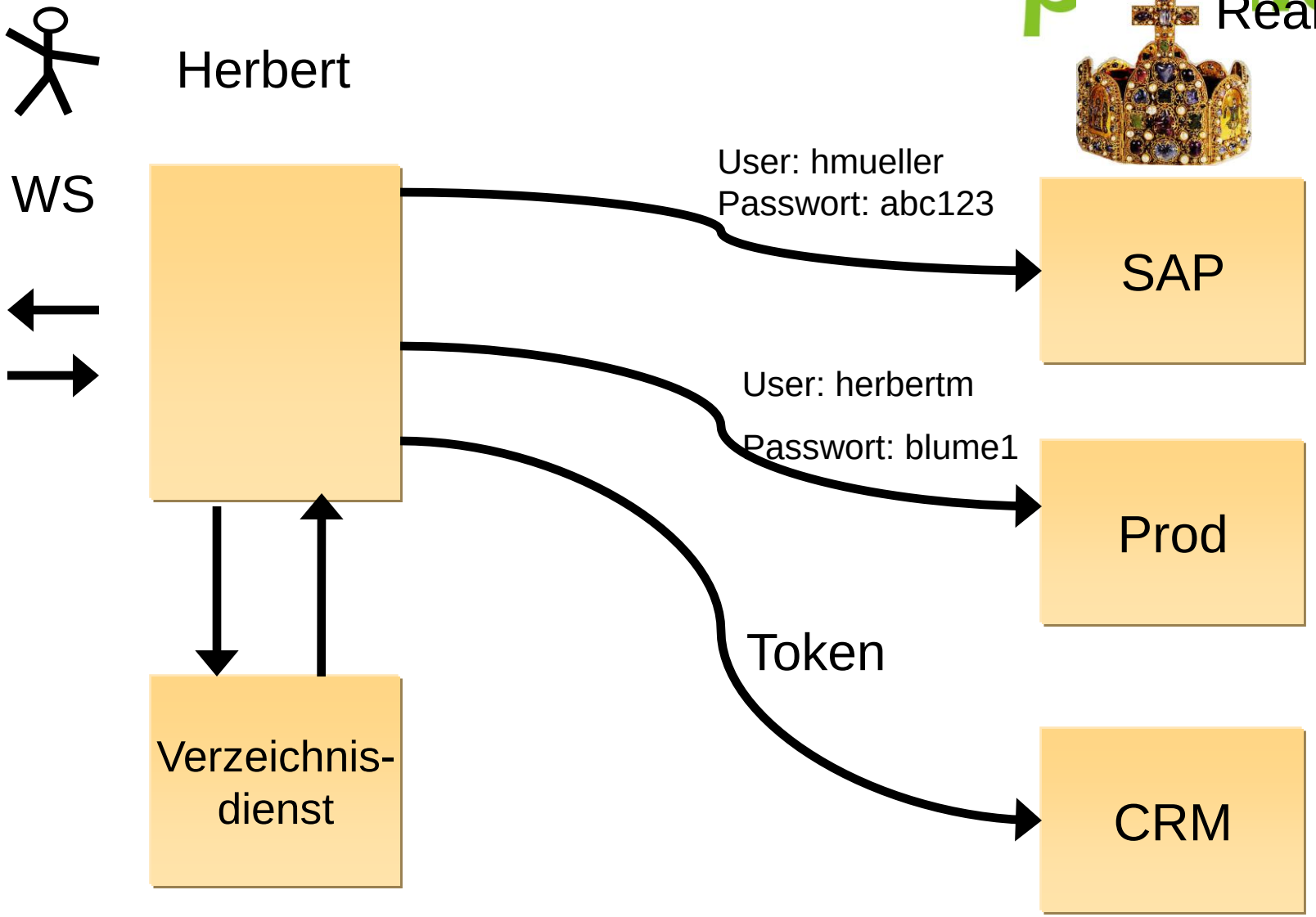
```
<sp:SignedSupportingTokens>
        <wsp:Policy>
          <sp:UsernameToken
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypol
icy/IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
              <sp:WssUsernameToken10 />
            </wsp:Policy>
          </sp:UsernameToken>
        </wsp:Policy>
      </sp:SignedSupportingTokens>
```

Single Sign On

Der tägliche Wahnsinn

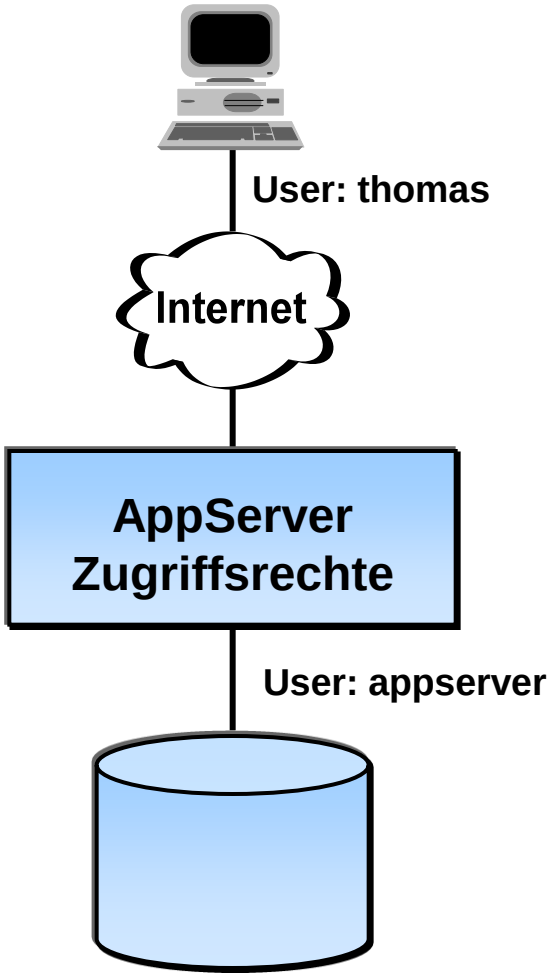


SSO mit Verzeichnisdienst

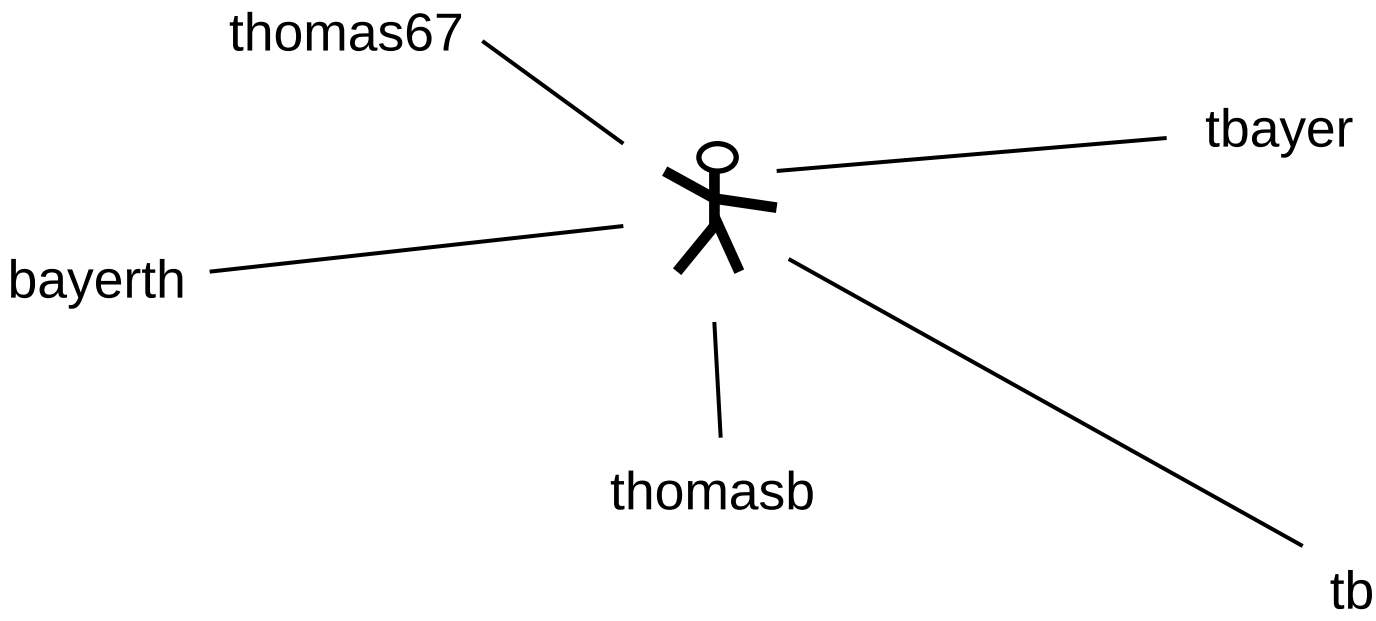


SSO mit Technischem User

- Einfache Realisierung
- Lizenzeinsparungen können möglich sein
- Keine feingranularen Zugriffsrechte
- Auditing ist unzureichend

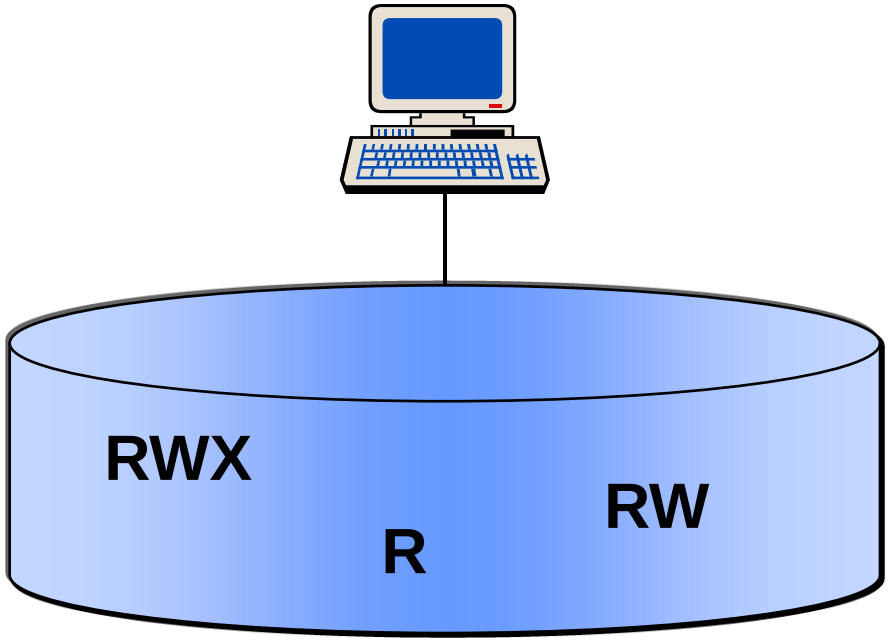


- Identitäten eines Benutzers in allen Systemen verwalten
- Single Point of Administration



Provisioning

- Zentrale Verwaltung der Benutzerrechte
- Single Point of Administration



Service Provisioning Markup Language SPML

- OASIS Standard
- Syntax und Protokolle für Provisioning
- TC Mitglieder
  - CA, SAP, Sun, …
- Transport über SOAP

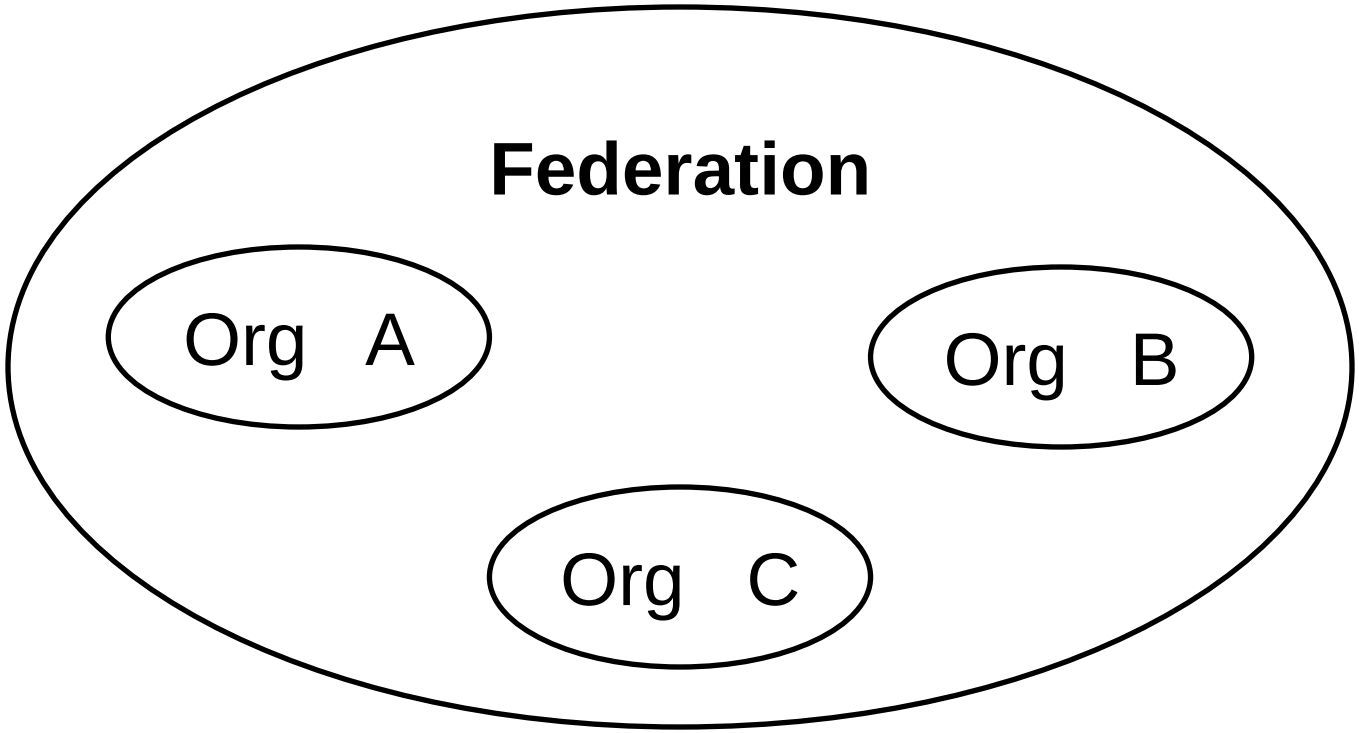
Privilege Management Infrastruktur

Software, Hardware, Personen, Prozesse und Richtlinien für die Erzeugung, Speicherung, Verteilung und Stornierung von Attribut Zertifikaten

Federations

Förderation / Federation

- Zusammenschluss von Organisationen
- Gemeinsame Richtlinien
- Vertrauen zwischen den Organisationen und deren Benutzern



- Problem: Datenschutz (z.B. EG-Recht)

Federation

- Assoziation von Service- und Identity Providern
- Vereinbarung von
  - Geschäftsbedingungen
  - Vertrauen
  - Benutzernamen
  - Attribute

  über die Grenzen mehrerer Organisationen

Richtlinien einer Federation

- Mitgliedschaft
- Aufnahme, Ausscheiden
- Rechte, Pflichten der Mitglieder
- Zugriff und Verwaltung

Liberty Alliance Project

- Develops open specifications for secure federated identity management
- More than 160 members
  - Educational, governmental, commercial
  - AOL, Ericsson, GM, HP, Nokia, Novell, NTT DoCoMo, Sony, Sun

Identity Federation Framework (ID-FF)

- Based on SAML
- From Liberty Alliance

Security Assertion Markup Language SAML

Security Assertion Markup Language SAML

- OASIS Standard
- Beschreibt Austausch von Authentifizierungs- und Authorisierungsinformationen
- Wird für SSO verwendet
- Sicherheitsinformationen „reisen“ mit dem Client

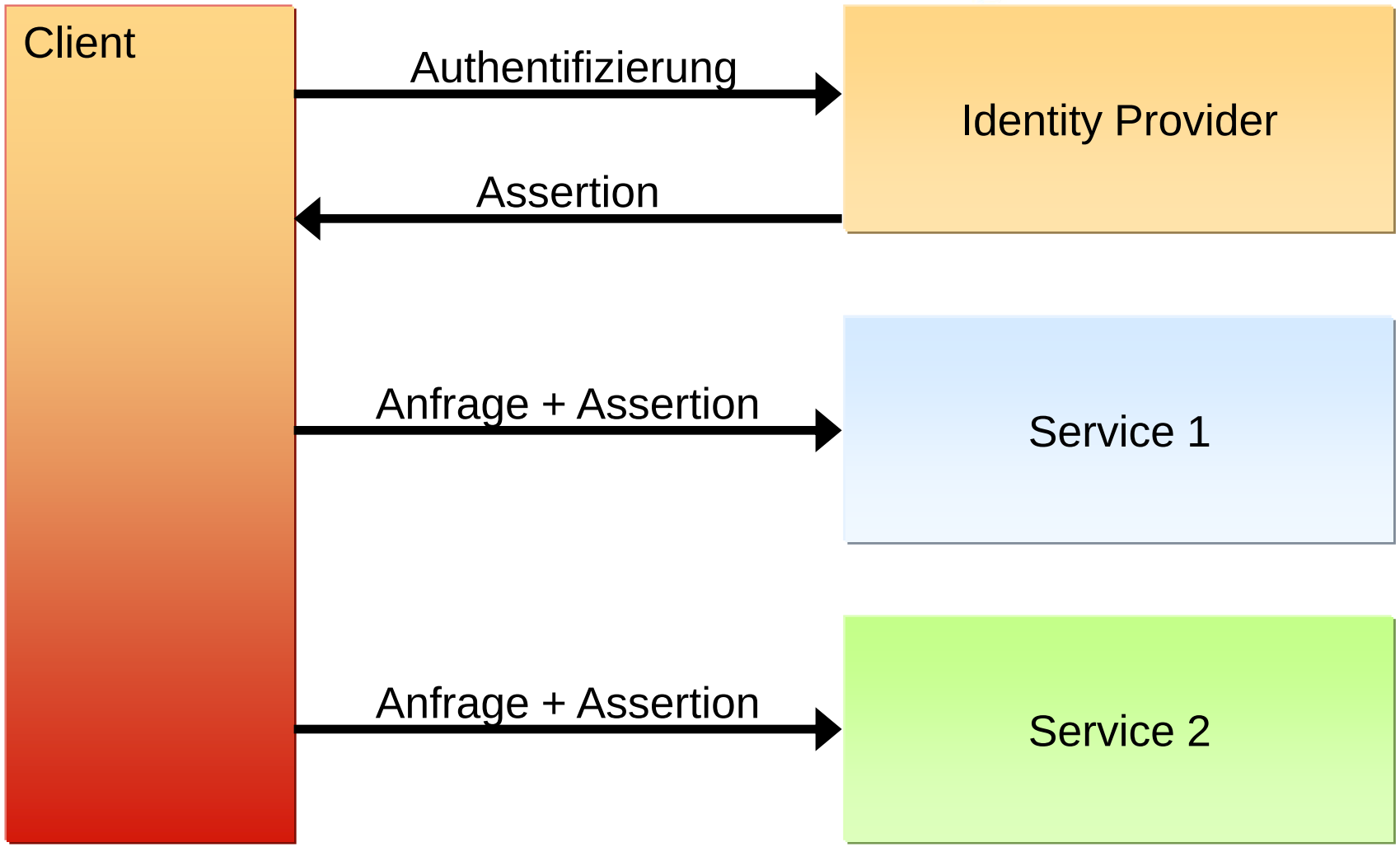
Assertion

- Aussage einer Authorität (Identity Provider) über einen Principal
- Inhalt
  - Herausgeber, Signatur, Subjekt, Bedingungen, Statements
- SAML Statements
  - Authentication
  - Attribute
  - Authorization Decision
- Wird von Identity zum Service Provider gesendet
- Können in WSS „eingewickelt„ werden

SAML Assertion (vereinfacht)

```
<env:Header>
  <wsse:Security>
    <saml:Assertion AssertionID="sfad86e51"
     IssueInstant="2007-01-15T16:47:19Z"
     Issuer="localhost">
      <saml:AuthenticationStatement
       AuthenticationInstant="2007-01-15T16:42:24Z"
       AuthenticationMethod="urn:com:sun:identity:Application">
        <saml:Subject>
          <saml:NameIdentifier>stockservice</saml:NameIdentifier>
          <saml:SubjectConfirmation>
            <saml:ConfirmationMethod>...:holder-of-key</saml:Confi
          </saml:SubjectConfirmation>
        </saml:Subject>
    </saml:AuthenticationStatement>
    <Signature><Reference URI="#sfad8 “/></Signature>
    </saml:Assertion>
  <wsu:TimeStamp>...</wsu:TimeStamp>
  </wsse:Security>
</env:Header>
```

SSO mit SAML: Assertions reisen mit dem Cient



Authentication Statement

- „Asserts“ dass der Prinzipal sich beim Identity Provider authentifiziert hat

Identity Provider

- Ein Principal registriert sich beim Identity Provider
- Generiert Assertions
- Beantwortet Authentifizierungsanfragen von Service Providern
- Art von STS

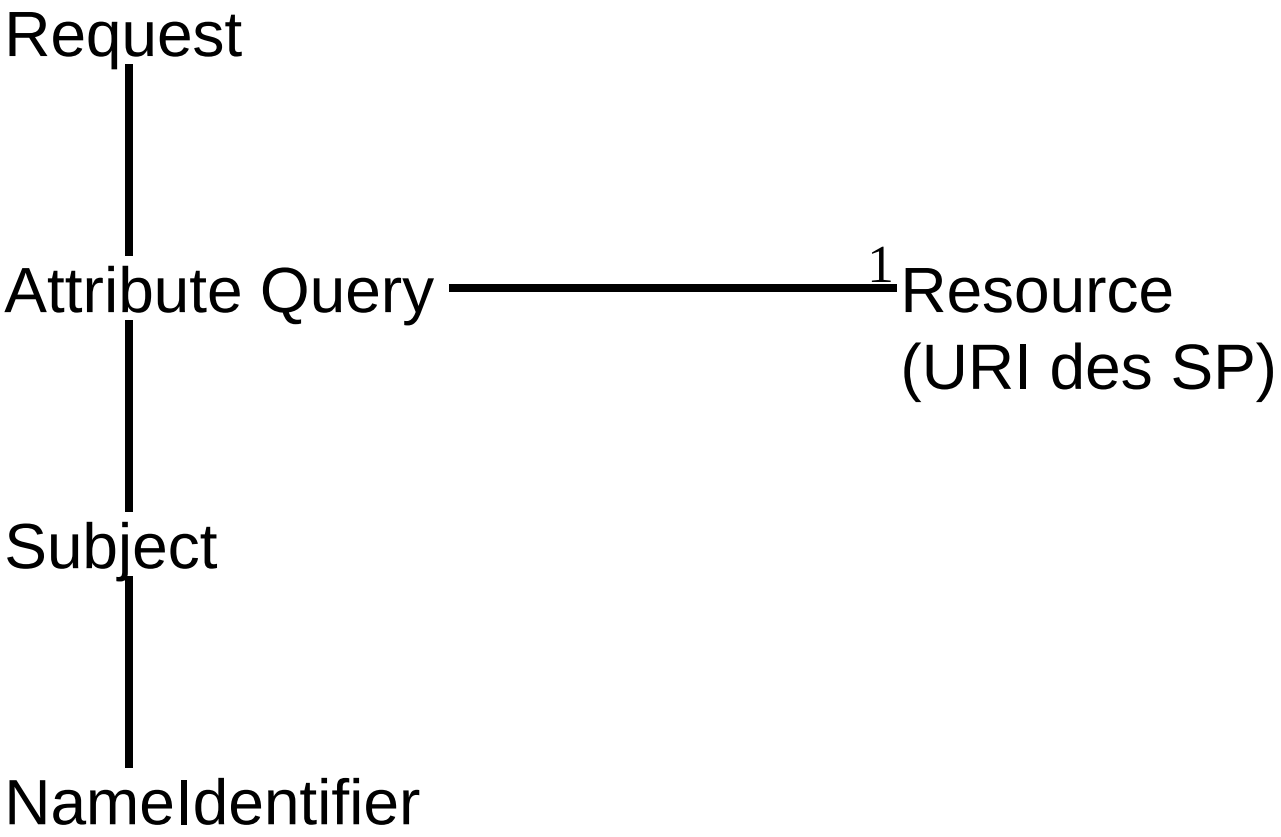
Service Provider

- Stellt Dienst zur Verfügung
- Trifft Entscheidung über Zugriff auf der Basis von Assertions
- Vertraut Identity Provider
- Kann Bestandteil einer Federation sein

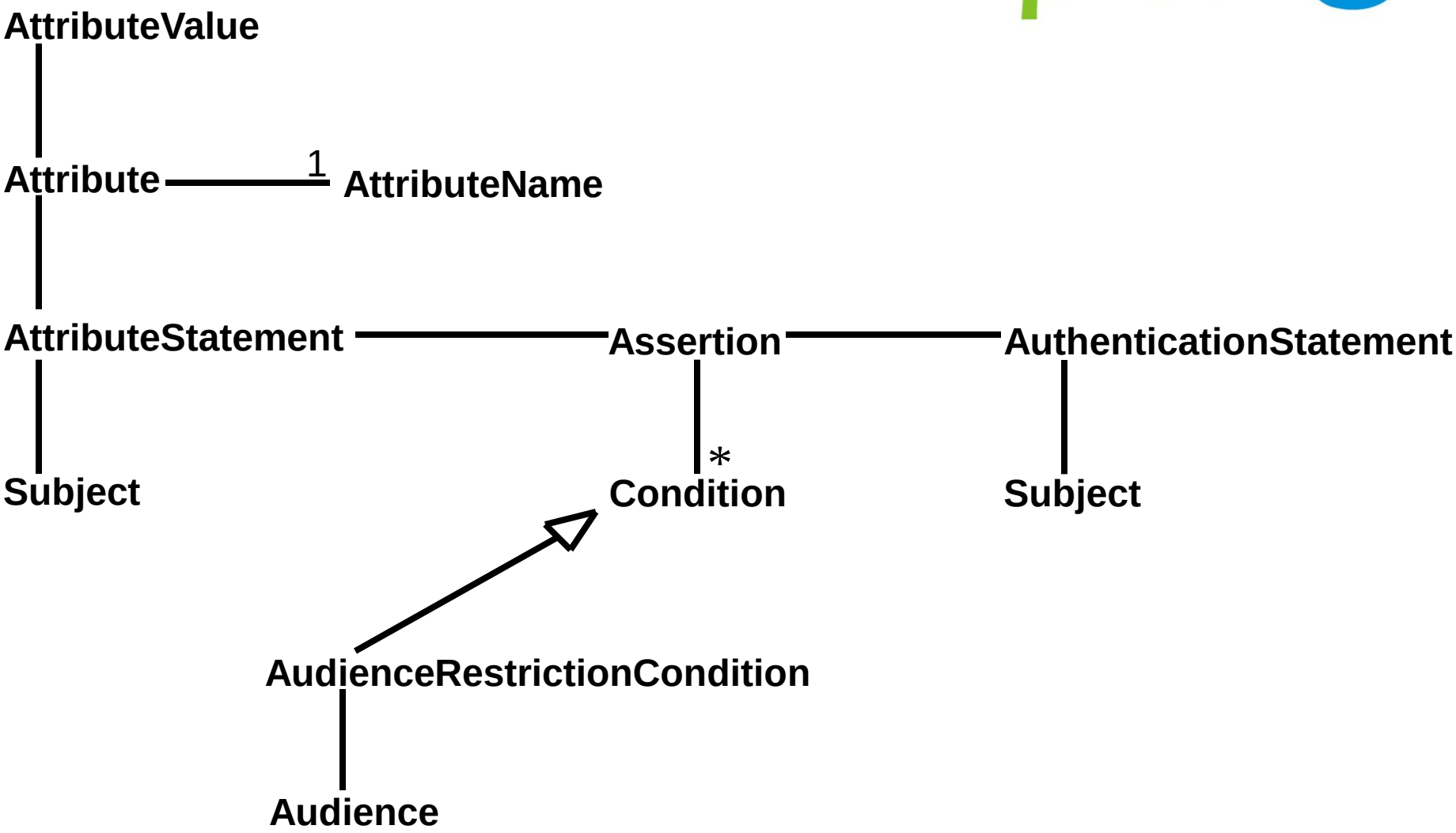
Inter-site Transfer Service

- Für den Übergang von einer Organisation zu einer Anderen
- Muss von einem Identity Provider angeboten werden

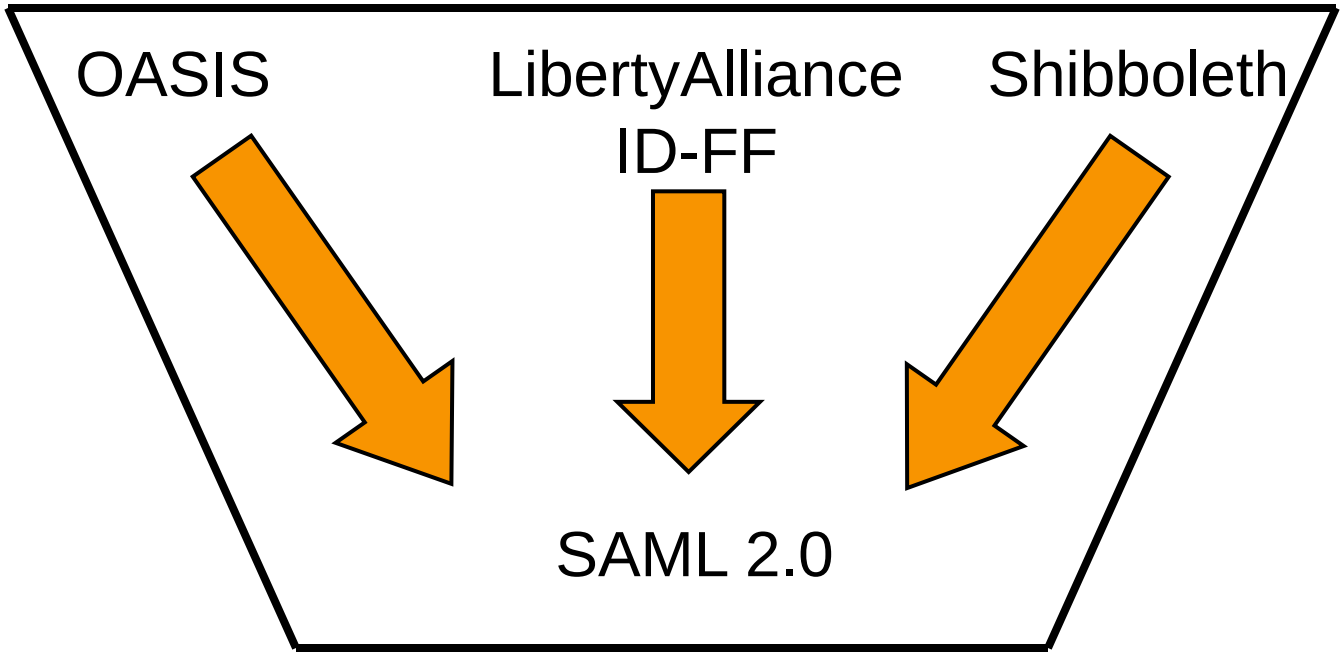
SAML Attributanfrage



SAML Assertion



SAML 2.0 Einflüsse



SAML 2.0

- Wurde durch Funktionen von Liberty ID-FF und Shibboleth beeinflußt
- Pseudonyme für Privacy
- Identifier Management (für Pseudonyme)
- Metadata
- Encryption
- Attribute Profile
- Session Management
- Unterstützung für mobile Geräte
- Identity Provider Discovery

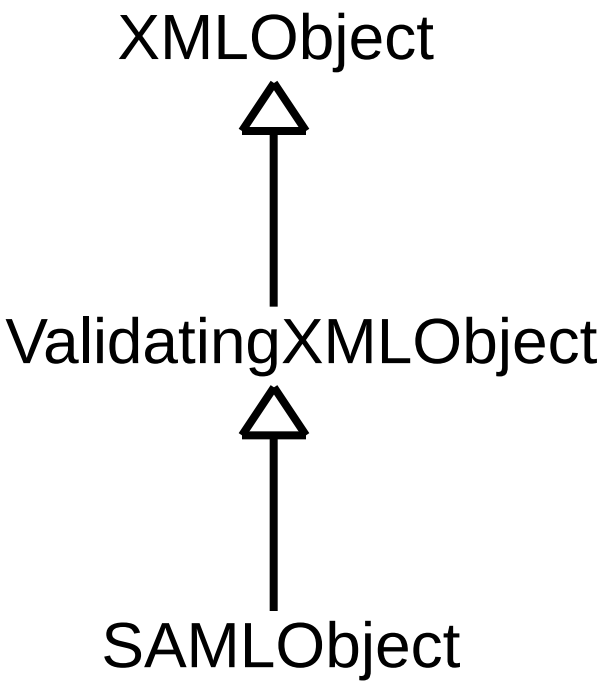
!OpenSAML

- Toolkit bzw. Bibliothek (JAR)
- Open Source unter ASF 2.0
- Von Internet2
- Infrastrukturprojekt von Shibboleth
- Java und C++
- Objektmodell für SAML
- Unterstützt SOAP Binding

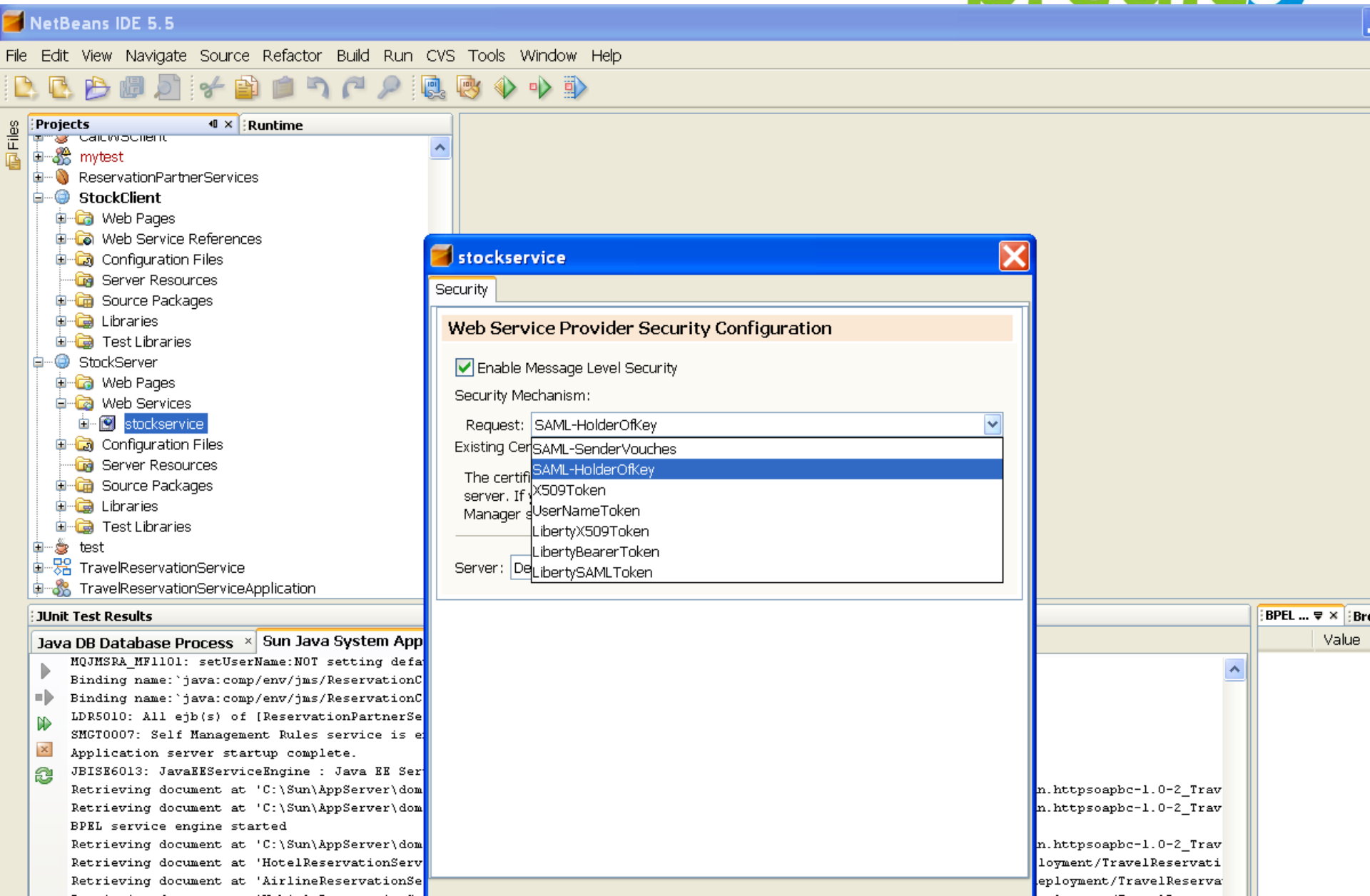
!Open SAML Versions

- 1.0
  - Unterstützt SAML 1.0, 1.1
- 2.0
  - In Entwicklung (Stand Okt. 07)
  - Wird SAML 1.0, 1.1 und 2.0 unterstützen
  - Komplette Neuentwicklung

!Open SAML Objekthierarchie (Auszug)



SAML Konfiguration in NetBeans



SSO mit Shibboleth

Buch der Richter, Kap. 12, Vers 5 und 6

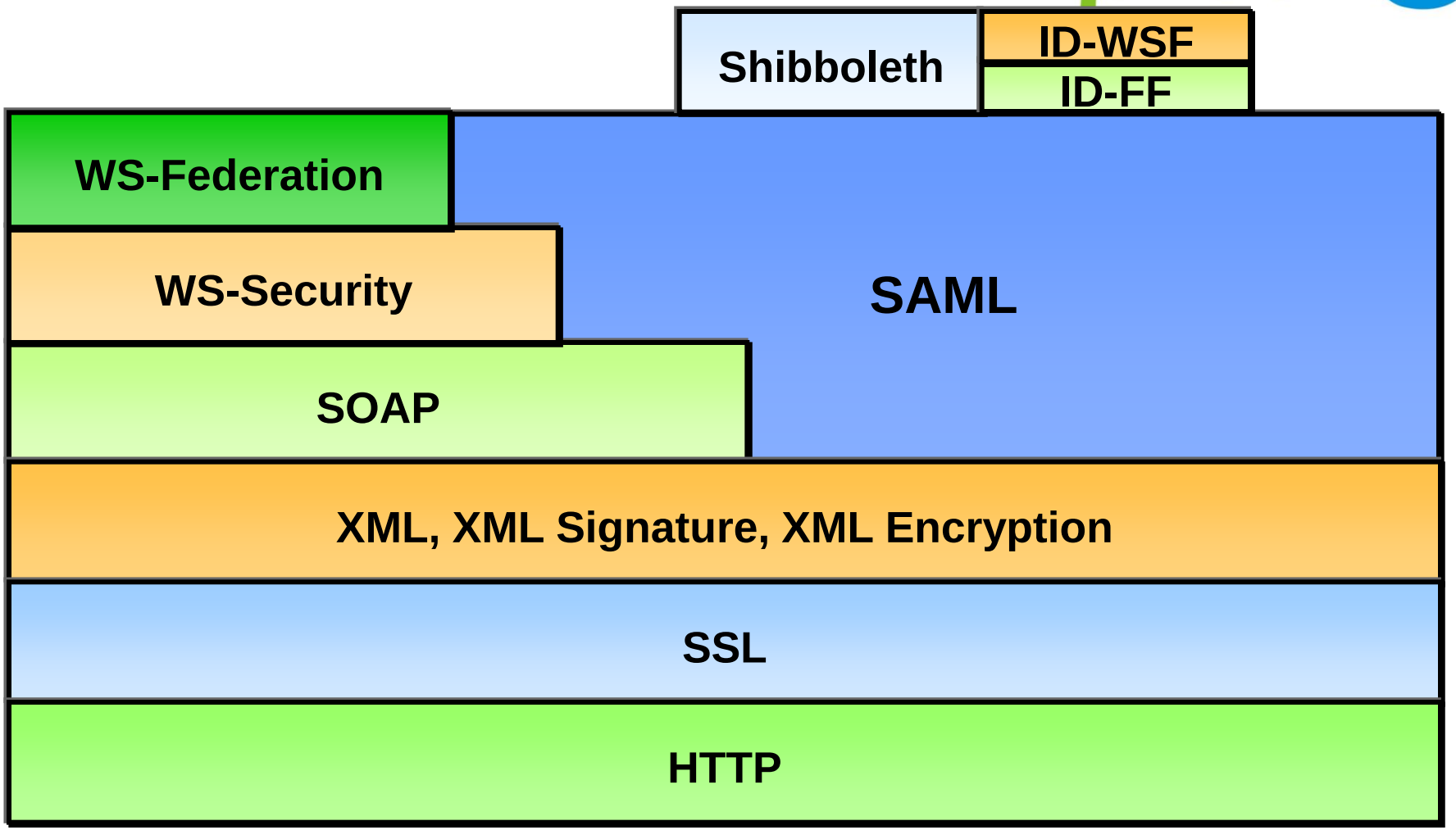
„Und die Gileaditer nahmen ein die Furten des Jordans vor Ephraim. Wenn nun die Flüchtigen Ephraims sprachen: Laß mich hinübergehen! so sprachen die Männer von Gilead zu Ihm: Bist du ein Ephraimiter? Wenn er dann antwortete: Nein!
hießen sie ihn sprechen: Schiboleth; so sprach er Siboleth und konnte es nicht recht reden; alsdann griffen sie ihn schlugen ihn an den Furten des Jordans, daß zu der Zeit von Ephraim fielen zweiundvierzigtausend.“

http://www.lateinwiki.org/index.php/Die_Bibel_-_Buch_der_Richter%2C_Kapitel_12

Shibboleth

- Projekt von Internet2 / MACE
- Ziel: Zugriffsschutz über Web Resourcen, die über Institutionen verteilt sind

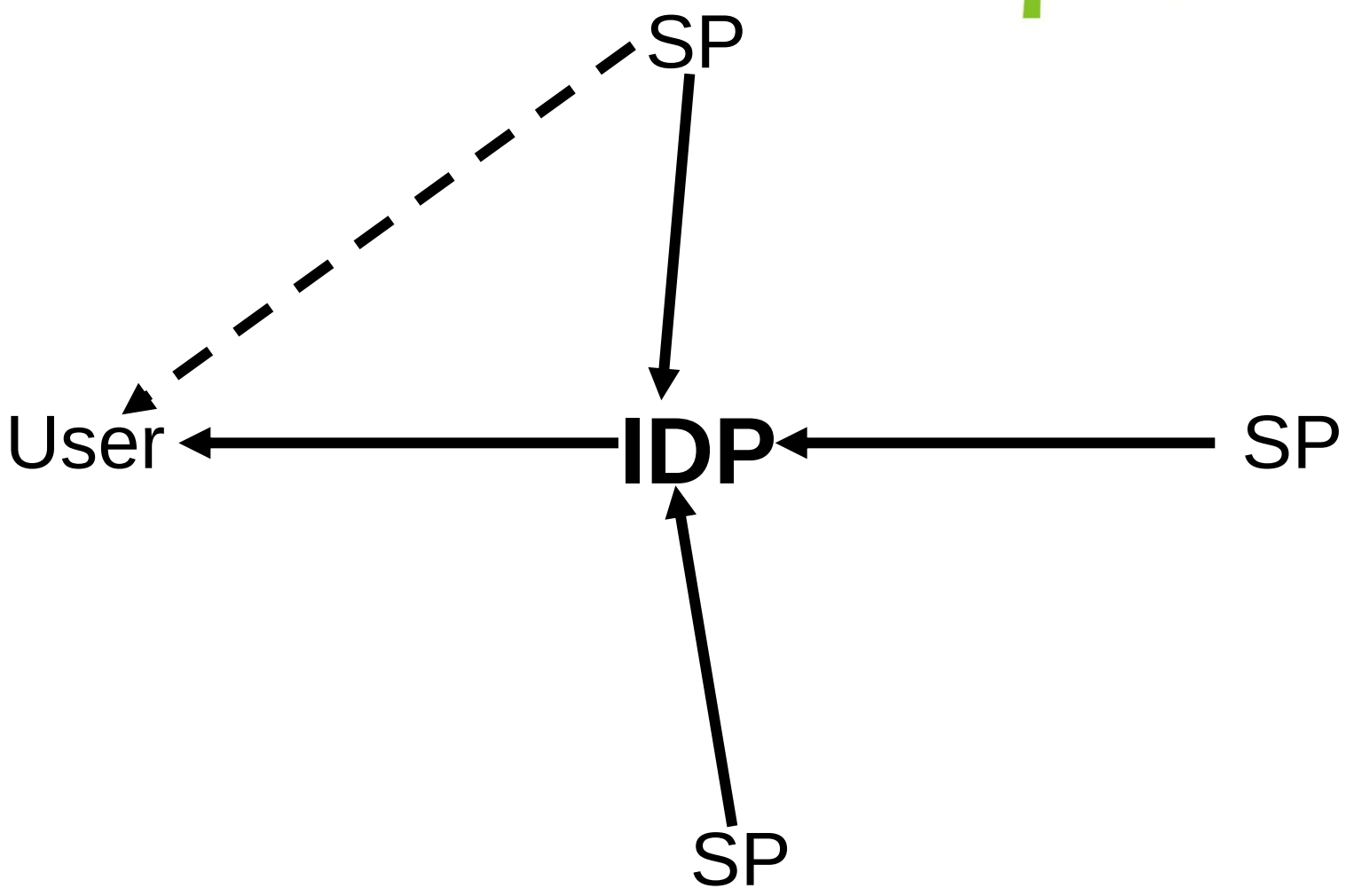
Web Services Security Stack



Schlüsselkonzepte von Shibboleth

- Föderale Administration
- Access Control
- Aktive Kontrolle der Privatsphäre
- Basierend auf Standards
- Framework für Vertrauen
- Standard Attribute

Vertrauen



Deine Freunde sind meine Freunde!

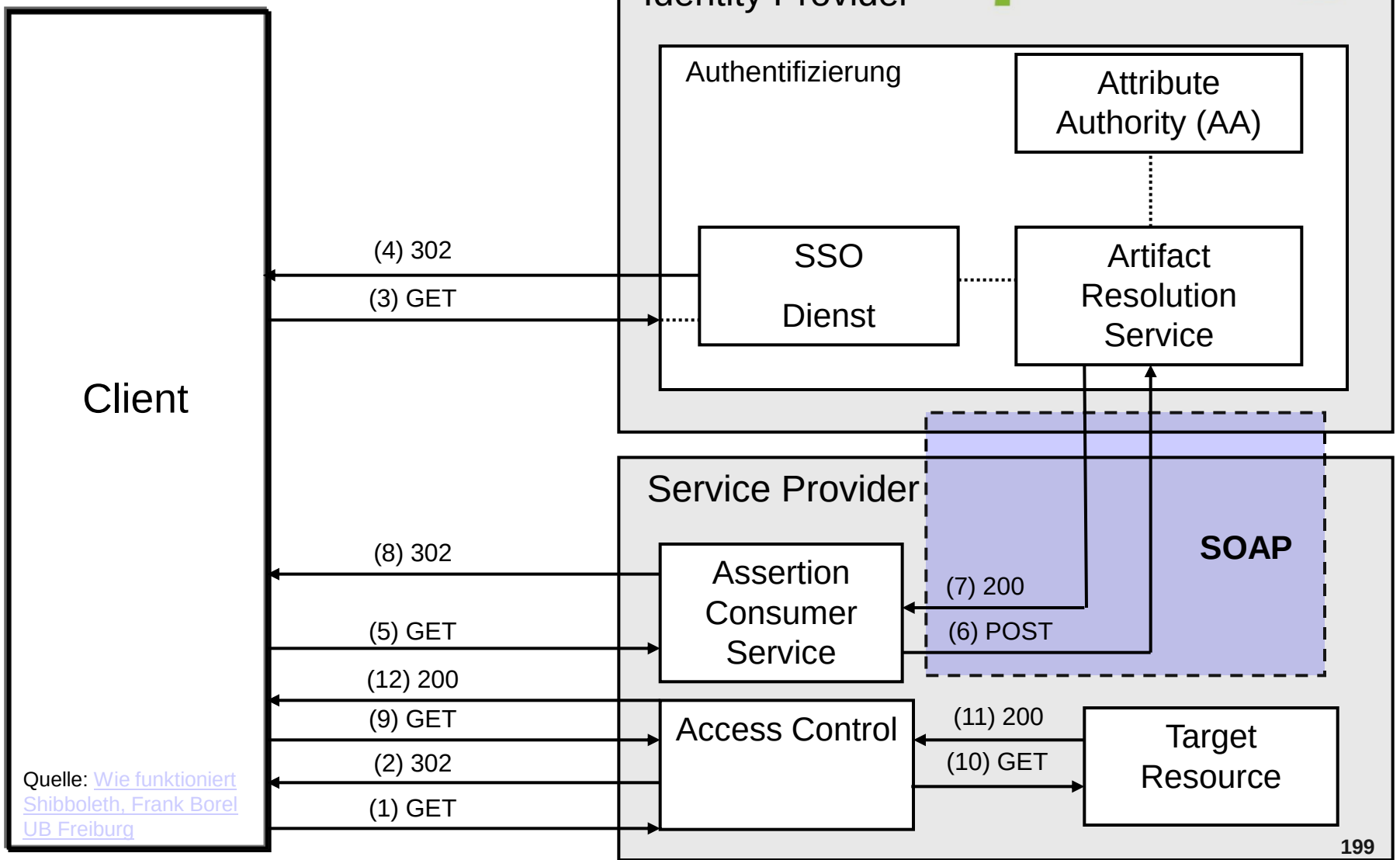
Shibboleth Trust Modell

- Annahmen/Anforderungen:
  - Sicherer Transport
  - Richtige Auslegung von Attributen
  - Verteilung von Zertifikaten
  - Digitale Unterschriften
  - Interoperabilität

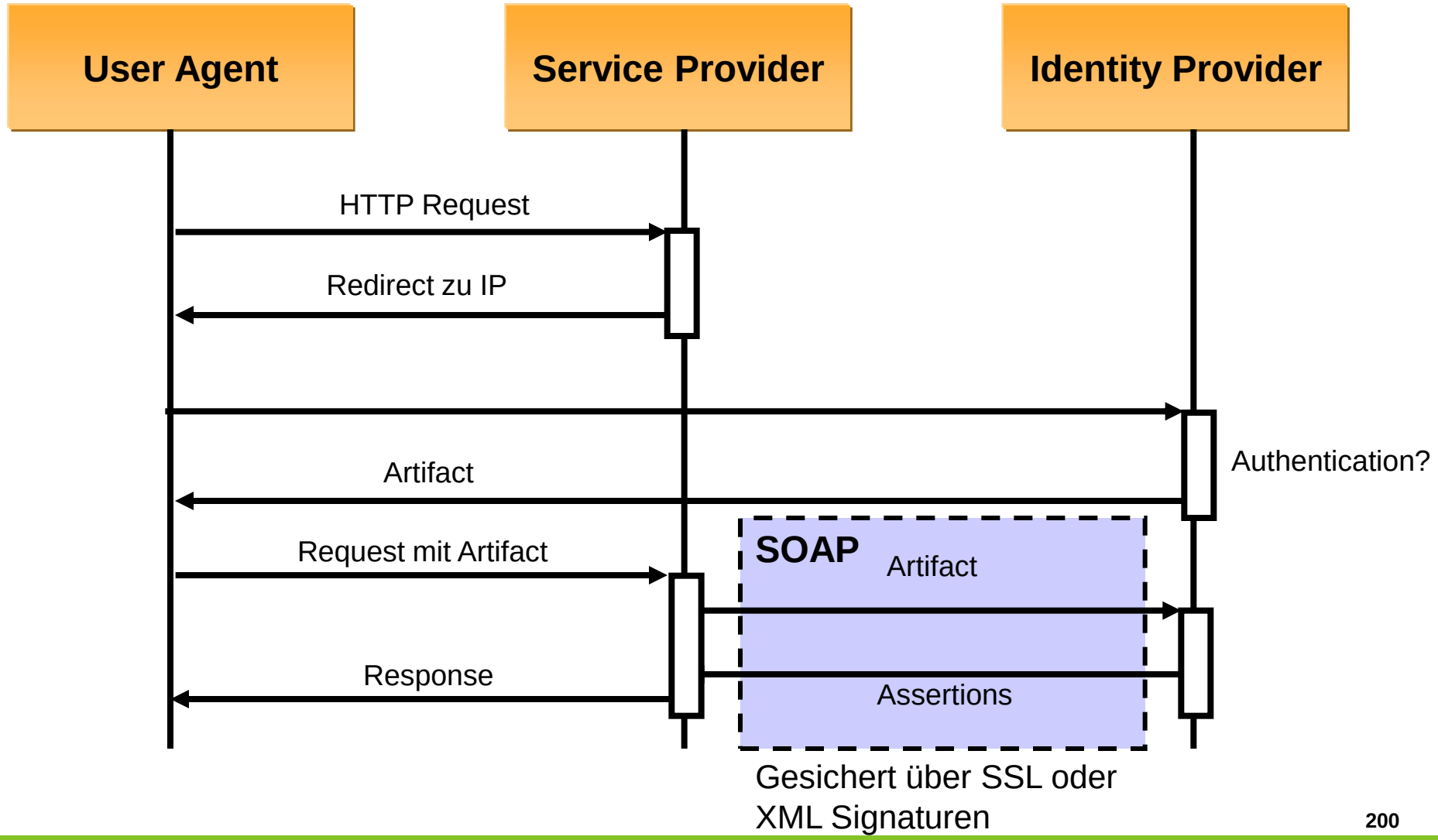
Browser /POST Profile

- SAML Response wird mit POST Operation an SP geschickt
- Verschiedene Artifakt-Formate

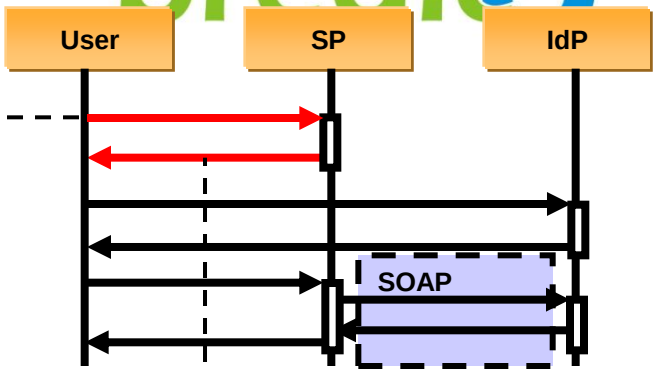
Erstkontakt mit Browser/Artifact



SSO über Browser/Artifact



Client greift auf vom SP geschützte Ressource zu



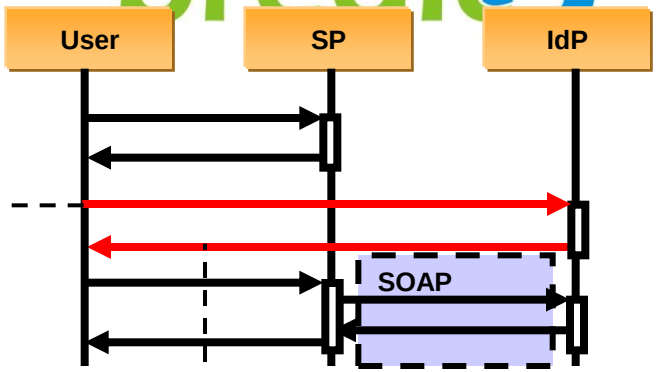
GET /secure/ HTTP/1.1
Host: sp.example.org:8080
User-Agent: Mozilla/5.0

Client

HTTP/1.x 302 Found
Server: Apache/2.2.6 (Win32) mod_jk/1.2.25
Set-Cookie:
_shibstate_1500711297c718afd85784f838e4305b6b568c9b=http%3A%2F%2Fsp.example.org%3A8080%2Fsecure%2F;
path=/
Location: https://idp.example.org:8443/shibboleth-idp/SSO?shire=http%3A%2F%2Fsp.example.org%3A8080%2FShibboleth.sso%2FSAML%2FArtifact&time=1191662750&target=cookie&providerId=https%3A%2F%2Flocalhost%2Fshibboleth
Content-Length: 397
Content-Type: text/html; charset=iso-8859-1

Service Provider

Client kontaktiert IdP



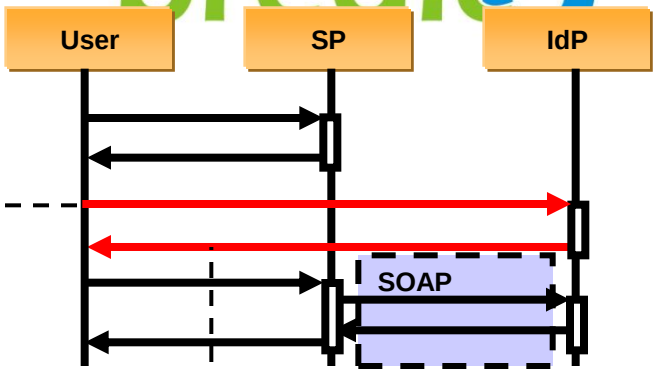
GET /shibboleth-idp/SSO?shire=http%3A%2F%2Fsp.example.org%3A8080%2FShibboleth.sso%2FSAML%2FArtifact&time=1191662750&target=cookie&providerId=https%3A%2F%2Flocalhost%2Fshibboleth HTTP/1.1
Host: idp.example.org:8443
User-Agent: Mozilla/5.0

Client

HTTP/1.x 401 Authorization Required
Server: Apache/2.2.6 (Win32) mod_ssl/2.2.6 OpenSSL/0.9.8e mod_jk/1.2.25
WWW-Authenticate: Basic realm="Villain Verification Service (VVS)"
Content-Length: 401
Content-Type: text/html; charset=iso-8859-1
---
https://idp.example.org:8443/shibboleth-idp/SSO?shire=http%3A%2F%2Fsp.example.org%3A8080%2FShibboleth.sso%2FSAML%2FArtifact&time=1191662750&target=cookie&providerId=https%3A%2F%2Flocalhost%2Fshibboleth

Identity Provider

Client authentifiziert sich beim IdP



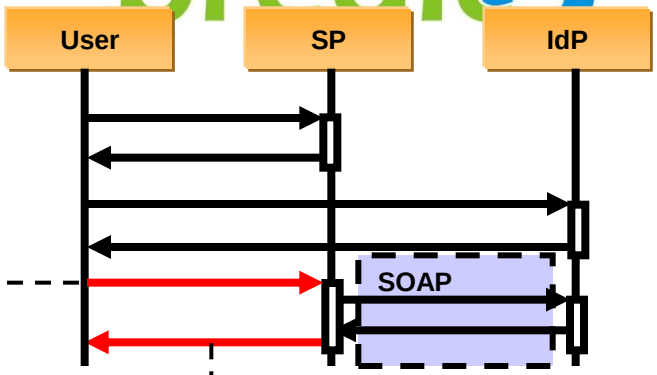
GET /shibboleth-idp/SSO?shire=http%3A%2F%2Fsp.example.org%3A8080%2FShibboleth.sso%2FSAML%2FArtifact&time=1191662750&target=cookie&providerId=https%3A%2F%2Flocalhost%2Fshibboleth HTTP/1.1
Host: idp.example.org:8443
User-Agent: Mozilla/5.0
Authorization: Basic bXlzZWxmOm15c2VsZg==

Client

HTTP/1.x 302 Moved Temporarily
Server: Apache/2.2.6 (Win32) mod_ssl/2.2.6 OpenSSL/0.9.8e mod_jk/1.2.25
Location:
http://sp.example.org:8080/Shibboleth.sso/SAML/Artifact?TARGET=cookie&SAMLart=AAH7iBsAkCvNPMBcQlDBx%2FAlFu8FW%2FaZHiyXfgndnEUOeAI2hi8Pzcko&SAMLart=AAH7iBsAkCvNPMBcQlDBx%2FAlFu8FWxlPGWmMZbMupKOgZxNPt8KstcRY

Identity Provider

Client übergibt Artifakt an SP



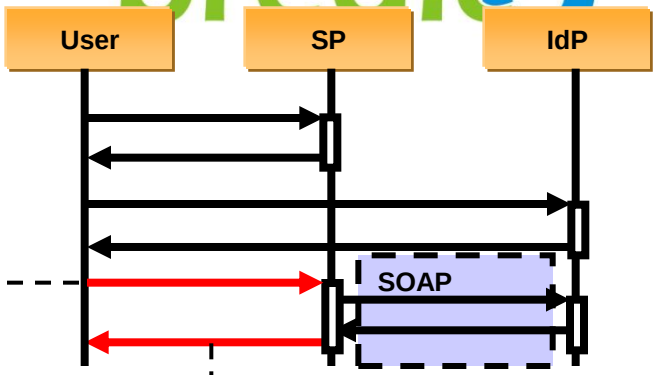
GET
/Shibboleth.sso/SAML/Artifact?TARGET=cookie&SAMLart=AAH7iBsAkCvNPMBcQlDBx%2FAlFu8FW%2FaZHiyXfgndnEUOeAI2hi8Pzcko&SAMLart=AAH7iBsAkCvNPMBcQlDBx%2FAlFu8FWxIPGWmMZbMupKOgZxNPt8KstcRY HTTP/1.1
Host: sp.example.org:8080
User-Agent: Mozilla/5.0
Cookie:
_shibstate_1500711297c718afd85784f838e4305b6b568c9b=http%3A%2F%2Fsp.example.org%3A8080%2Fsecure%2F

Client

HTTP/1.x 302 Found
Server: Apache/2.2.6 (Win32) mod_jk/1.2.25
Set-Cookie: _shibsession_1500711297c718afd85784f838e4305b6b568c9b=_1e4f004c67659def94470586ef36419f; path=/
Set-Cookie: _saml_idp=aHR0cHM6Ly9pZHAuZXhhbXBsZS5vcmcvc2hpYmJvbGV0aA%3D%3D; path=/; expires=Sat, 13 Oct 2007 09:25:54 GMT
Location: http://sp.example.org:8080/secure/
Content-Length: 218
Content-Type: text/html; charset=iso-8859-1
---
http://sp.example.org:8080/secure/

Service Provider

Client greift erneut auf gesicherte Resource zu



GET /secure/ HTTP/1.1
Host: sp.example.org:8080
User-Agent: Mozilla/5.0
Cookie:
_shibstate_1500711297c718afd85784f838e4305b6b568c9b=http%3A%2F%2Fsp.example.org%3A8080%2Fsecure%2F;
_shibsession_1500711297c718afd85784f838e4305b6b568c9b=_1e4f004c67659def94470586ef36419f;
_saml_idp=aHR0cHM6Ly9pZHAuZXhhbXBsZS5vcmcvc2hpYmJvbGV0aA%3D%3D

Client

HTTP/1.x 200 OK
Server: Apache/2.2.6 (Win32) mod_jk/1.2.25
Content-Length: 41
Content-Type: text/html
---
http://sp.example.org:8080/favicon.ico

Service Provider

Shibboleth Targets

- Apache Modul
- Konfiguration
  - Zentrale Config Datei oder
  - Lokale .htaccess Dateien

Shibboleth NameIdentifier

- Identifier ist transient und opaque
- Gültiger SAML Identifier
- Name Qualifier enthält Identifier des IdP welcher den transienten Identifier generiert hat

Shibboleth Club

- Instanz einer förderierten Administration
- Stellen Richtlinien über Attribut-Verwendung der Service Provider auf

Where are you from? Service WAYF

- Liefert Identity Provider eines Principals
- Optional
- “Proxy” für Service Provider

Probleme

- Attribute Authorities sollten Zugriffsschutz haben
- Zeit Synchonisation (<5 min)
- User Privacy in Bezug auf SP Resourcen
  (Lösung: Transient, Identifiers, Opaque URIs,..)
- Ausfallsicherheit
- Was macht SP mit Attribut-Informationen?
- Firewalls

Shibboleth 2

- In Entwicklung (Stand 10/2007)
- Unterstützung eines Großteils von SAML 2.0
  - IdP und SP Lite conformance Sets sollen annähernd erfüllt werden
- Fokus: Notwendiges für Browser SSO und Single Logout
- SOAP Bindings
- Abwärtskompatibel (Wire/Runtime) zu Shib. 1.3
- Verschlüsselung von Zusicherungen
- Java SP als Servlet 2.4 Filter
- Vorhaben
  - Namen Identifier Management
  - Web Services Security WSS für SAML SOAP Requests

Bullshit Bingo

- Artifact: Referenz zu einer Authentication Assertion
- Attribute Push: Identity Provider liefert zusammen mit Athentifizierung Attribut Zusicherungen über den Principal

WS-Trust

WS-Trust

- OASIS Standard seit 19. März 2007
- Erweitert, WSS
- Ausstellen und Verteilen von Credentials über Domain Grenzen
- Aufbau von Vertrauen über Token
  - Ausgabe, Validierung und Erneuerung von Tokens
- Verteilung von Zertifikaten zur Laufzeit
- Basiert auf: WSS, WS-Addressing, WS-Policy

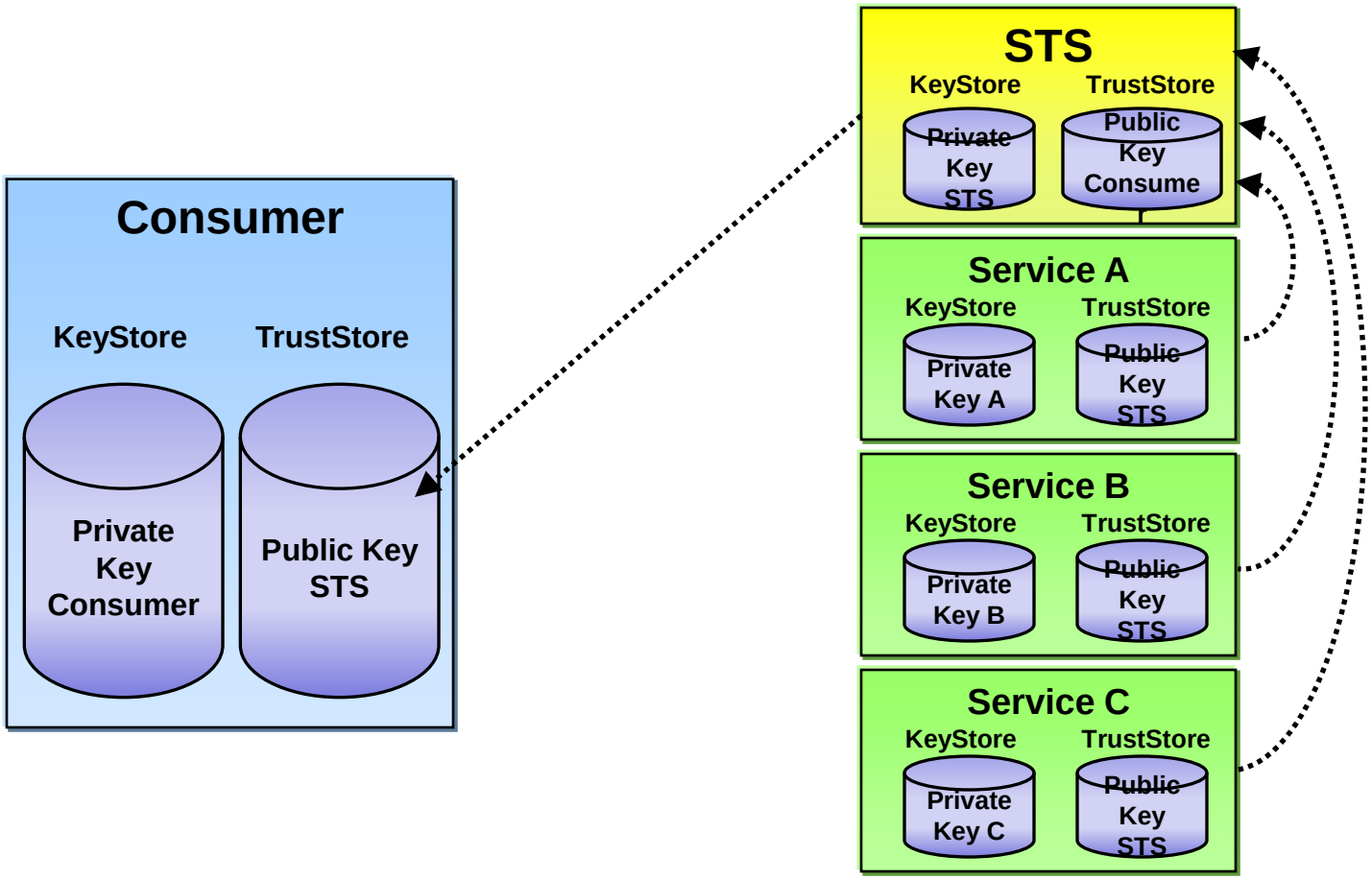
WS-Trust Non-Goals

- Authentifizierung über Passwörter
- Revokation von Token
- Trust Policy Management

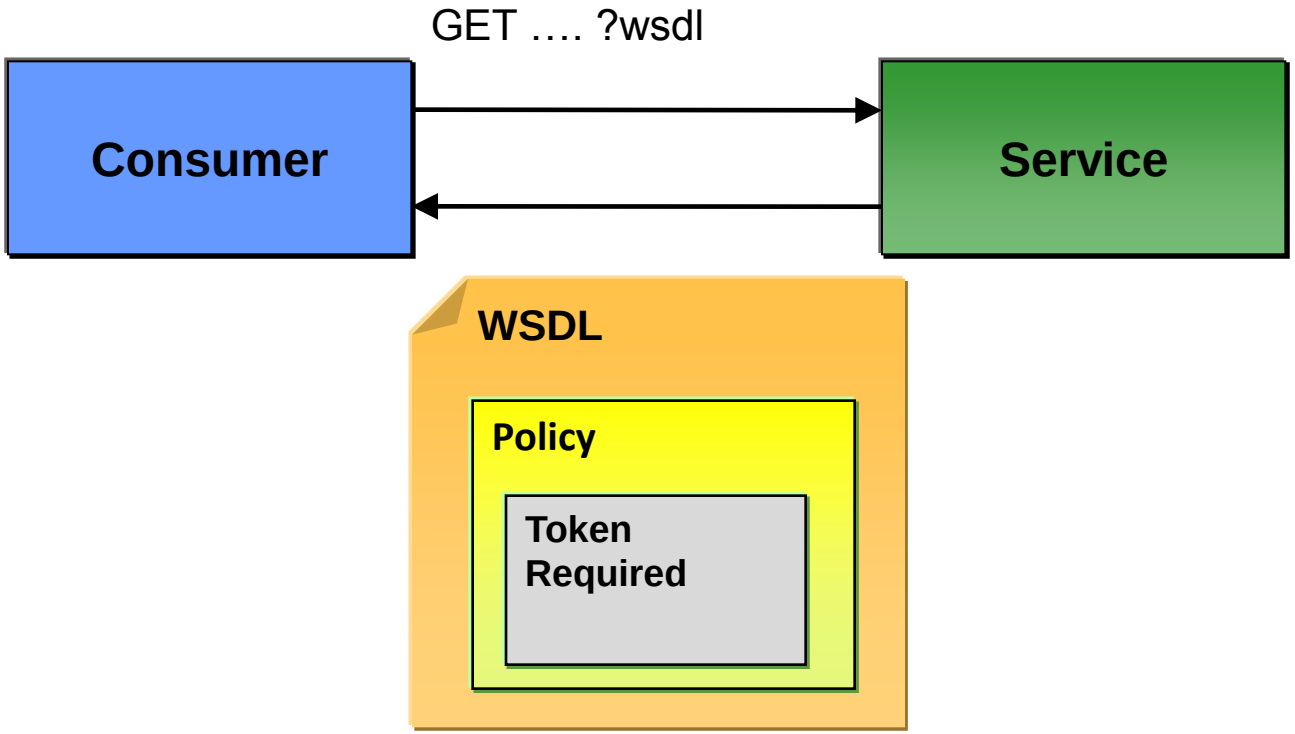
Trust Bootstrap

- Festgelegte vertrauenswürdige Root-Zertifikate
- Trust Hierarchies
- Vertrauenswürdiger Identity Dienst

WS-Trust Beispiel: Vertrauen



Consumer fragt nach WSDL Service



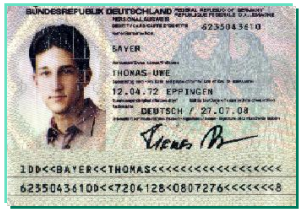
STS Configuration

```
<wsp:All>
  <sc:ValidatorConfiguration
   xmlns:sc="http://schemas.sun.com/2006/03/wss/server">
  </sc:ValidatorConfiguration>
  <wst:STSConfiguration encryptIssuedKey="true" encryptIssuedToken="false">
    <wst:LifeTime>36000</wst:LifeTime>
    <wst:Contract>
      com.sun.xml.ws.security.trust.impl.IssueSamlTokenContractImpl
    </wst:Contract>
    <wst:Issuer>SunSTS</wst:Issuer>
    <wst:ServiceProviders>
      <wst:ServiceProvider endpoint="default" >
        <wst:CertAlias>xws-security-server</wst:CertAlias>
        <wst:TokenType>…#SAMLV1.1</wst:TokenType>
      </wst:ServiceProvider>
    </wst:ServiceProviders>
  </wst:STSConfiguration>
  <wsap10:UsingAddressing/>
</wsp:All>
```

Policy in der WSDL des Service

```
<wssp:ProtectionToken>
  <wsp:Policy>
    <wsp:ExactlyOne>
      <wsp:All>
        <wssp:IssuedToken
wssp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
          <wsp:Policy>
            <wsp:ExactlyOne>
              <wsp:All>
                <ns4:RequireInternalReference />
              </wsp:All>
            </wsp:ExactlyOne>
          </wsp:Policy>
          <wssp:RequestSecurityTokenTemplate>
            <wst:KeySize>256</wst:KeySize>
            <wst:KeyType>http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey</wst:KeyType>
            <wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1</wst:TokenType>
          </wssp:RequestSecurityTokenTemplate>
        </wssp:IssuedToken>
      </wsp:All>
    </wsp:ExactlyOne>
  </wsp:Policy>
</wssp:ProtectionToken>
```

Authentifikation und Autorisierung über Token Service



②

STS

Token

Consumer

①

„Token benötigt“

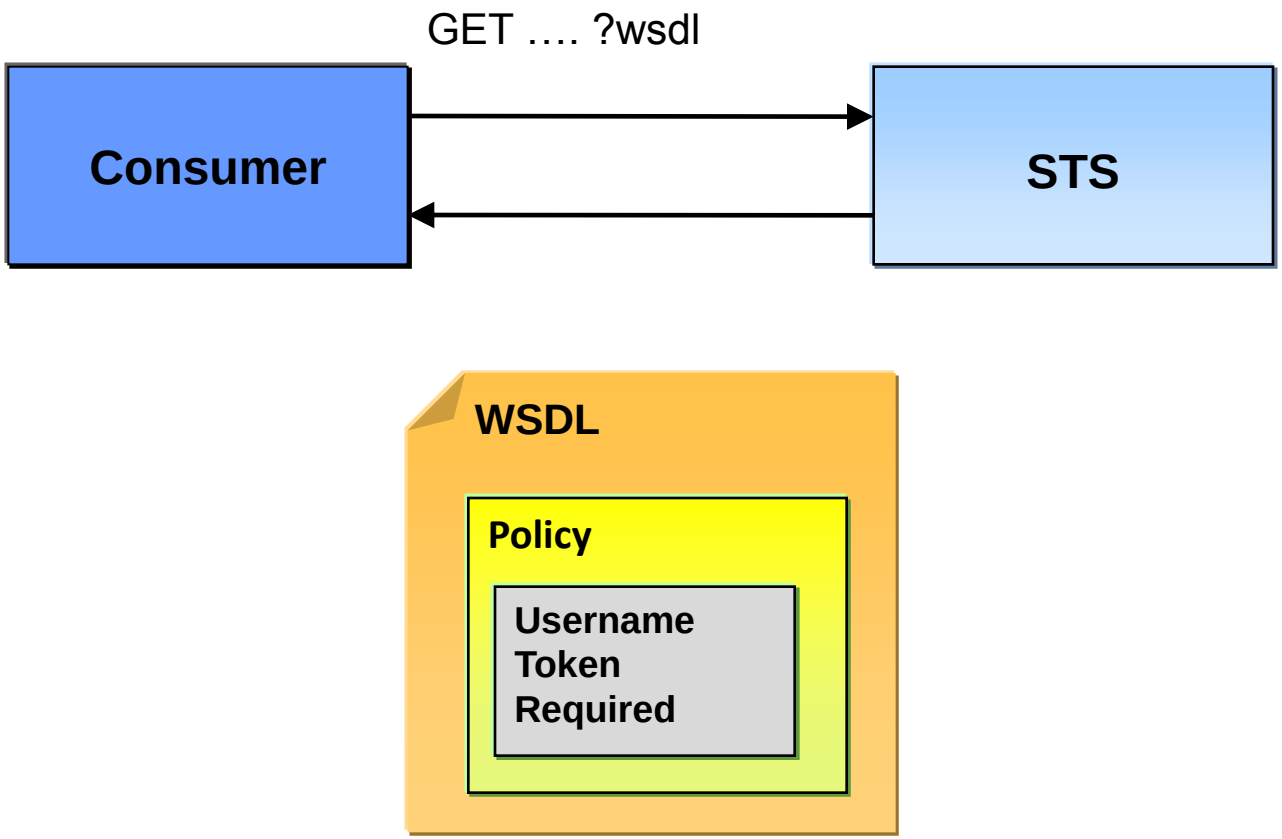
Service

③

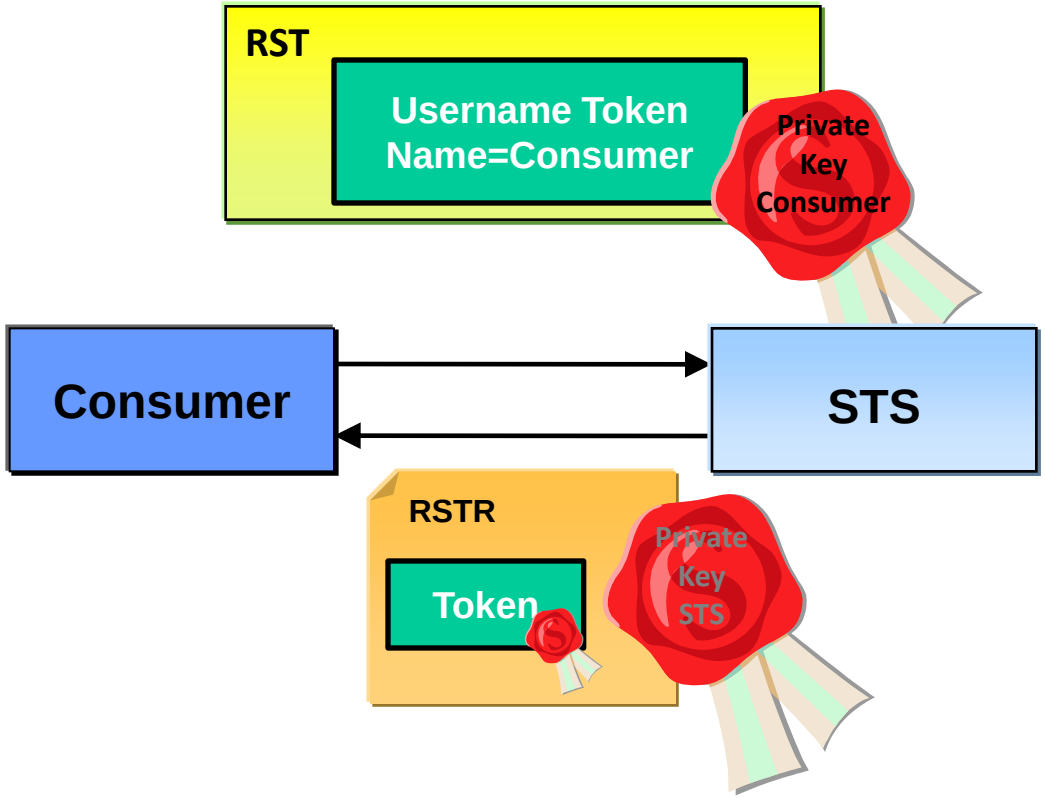
Token

④

Consumer fragt nach WSDL des STS



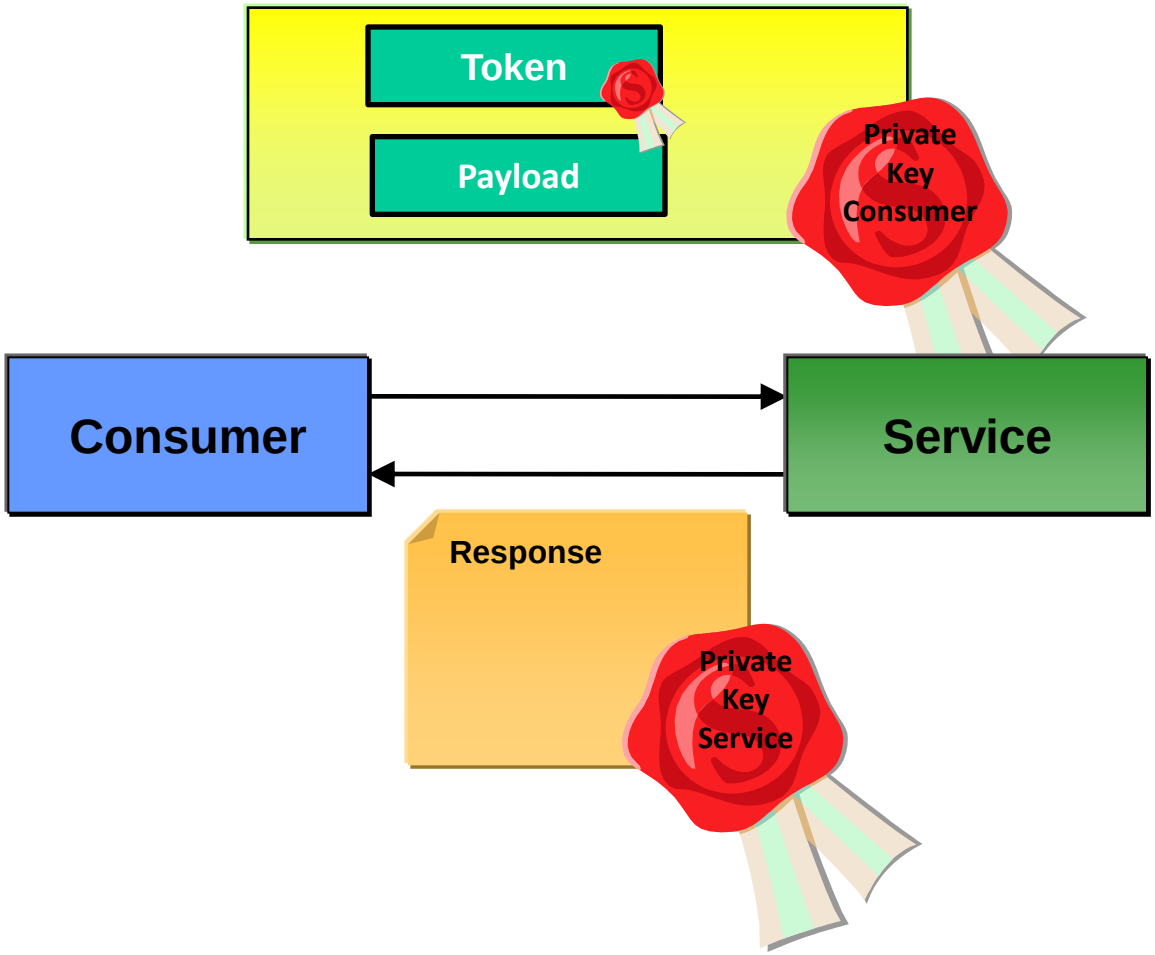
Consumer fordert Token beim STS an



Request Security Token (RST)

```
<RequestSecurityToken>
  <RequestType>
    http://schemas.xmlsoap.org/ws/2005/02/trust/Issue
  </RequestType>
  <AppliesTo>
    <EndpointReference>
      <Address>http://localhost:2000/payment/PaymentServiceService</Address>
    </EndpointReference>
  </AppliesTo>
  <TokenType>
    http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1
  </TokenType>
  <KeyType>
    http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey
  </KeyType>
  <KeySize>128</KeySize>
  <Entropy>
    <BinarySecret>LGNjP8PLVvuh==</BinarySecret>
  </Entropy>
  <ComputedKeyAlgorithm>
    http://schemas.xmlsoap.org/ws/2005/02/trust/CK/PSHA1
  </ComputedKeyAlgorithm>
</RequestSecurityToken>
```

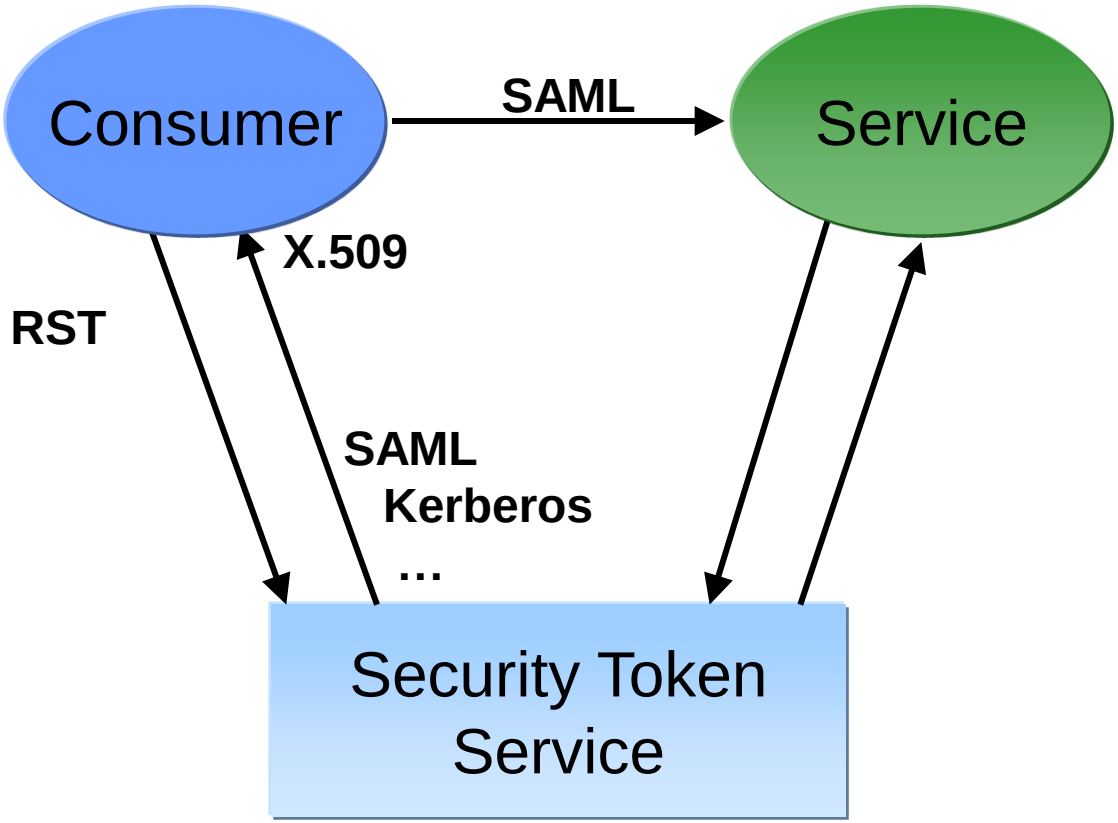
Consumer ruft Service auf



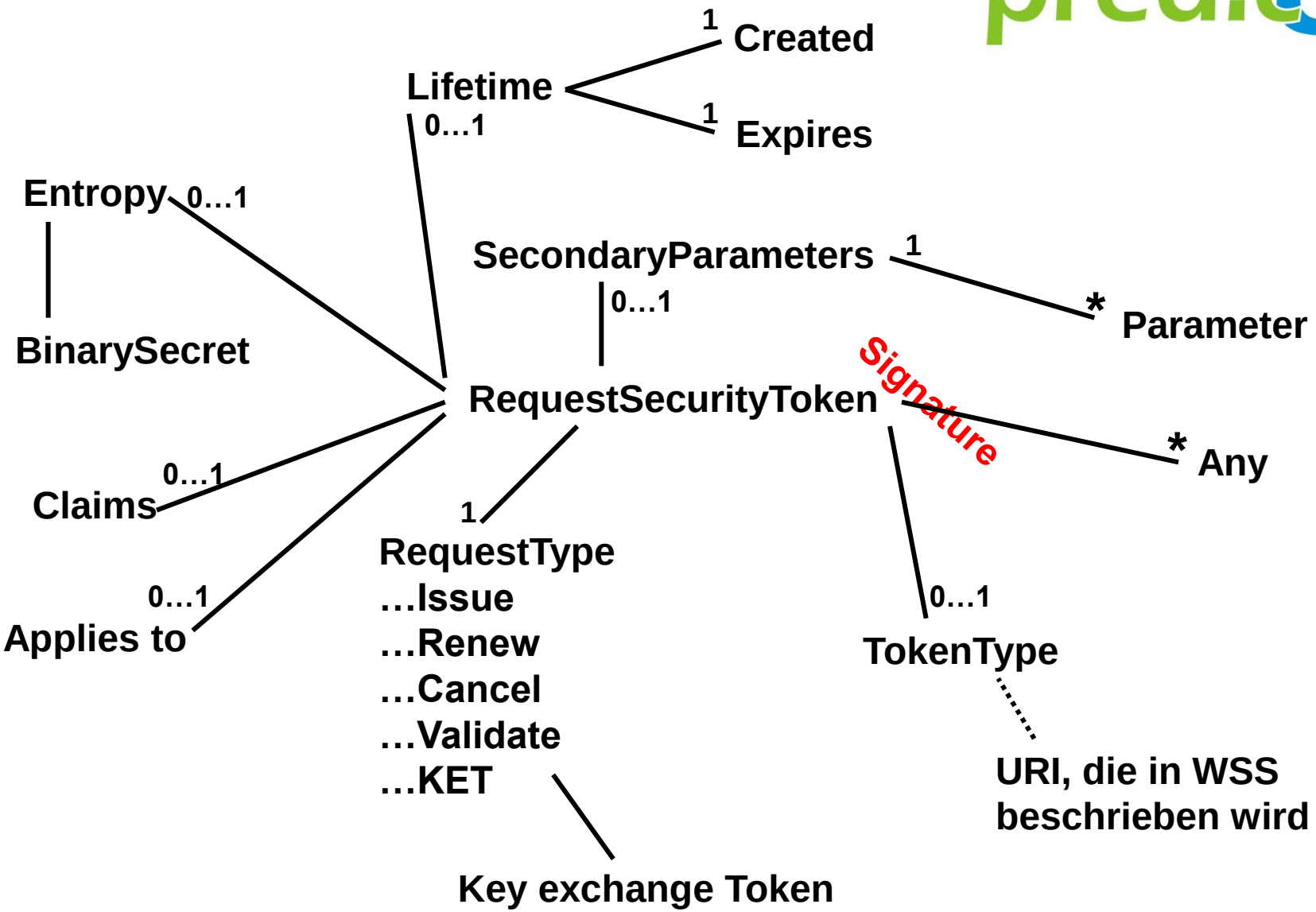
SAML Token

```
<saml:Assertion AssertionID="uuid-7bf" IssueInstant="2008-06-13T16:07:44.812Z"
                Issuer="SunSTS" MajorVersion="1" MinorVersion="1">
  <saml:Conditions NotBefore="2008-06-13T16:07:44.812Z"
                   NotOnOrAfter="2008-06-13T16:08:20.812Z"/>
  <saml:AttributeStatement>
    <saml:Subject>
      <saml:NameIdentifier NameQualifier="http://sun.com">
        thomas
      </saml:NameIdentifier>
      <saml:SubjectConfirmation>
        <saml:ConfirmationMethod>…:holder-of-key</saml:ConfirmationMethod>
        <ds:KeyInfo>…</ds:KeyInfo>
      </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Attribute AttributeName="token-requestor„
                    AttributeNamespace="http://sun.com">
      <saml:AttributeValue xsd:type="xsd:string">authenticated</saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>
```

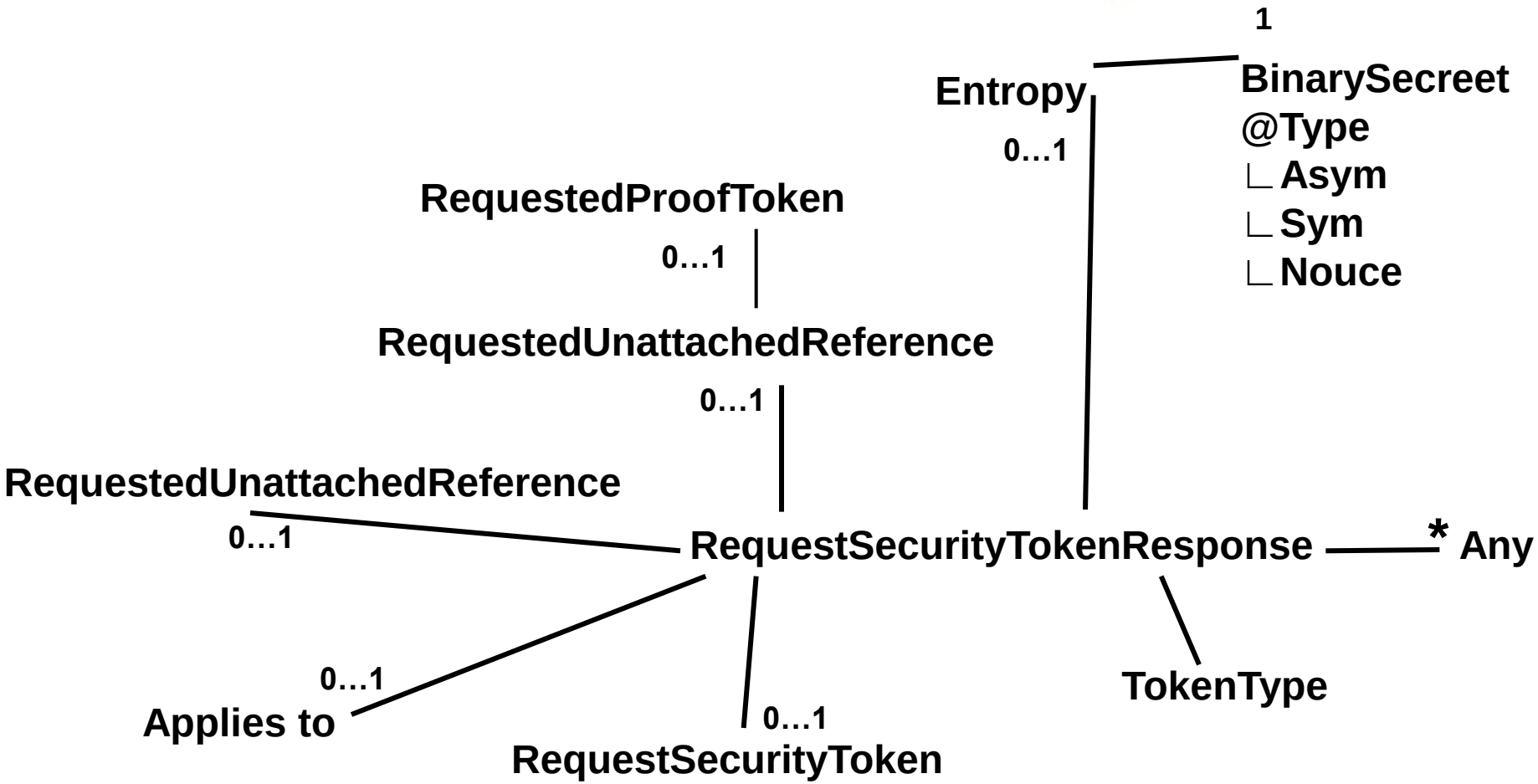
URL des Service-Zertifikat



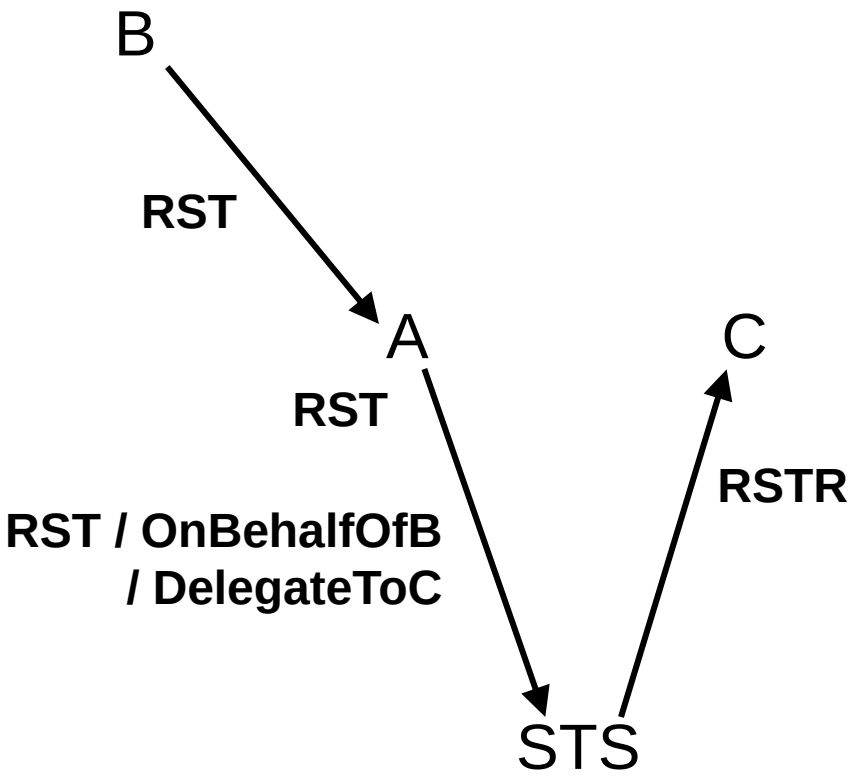
Struktur RST



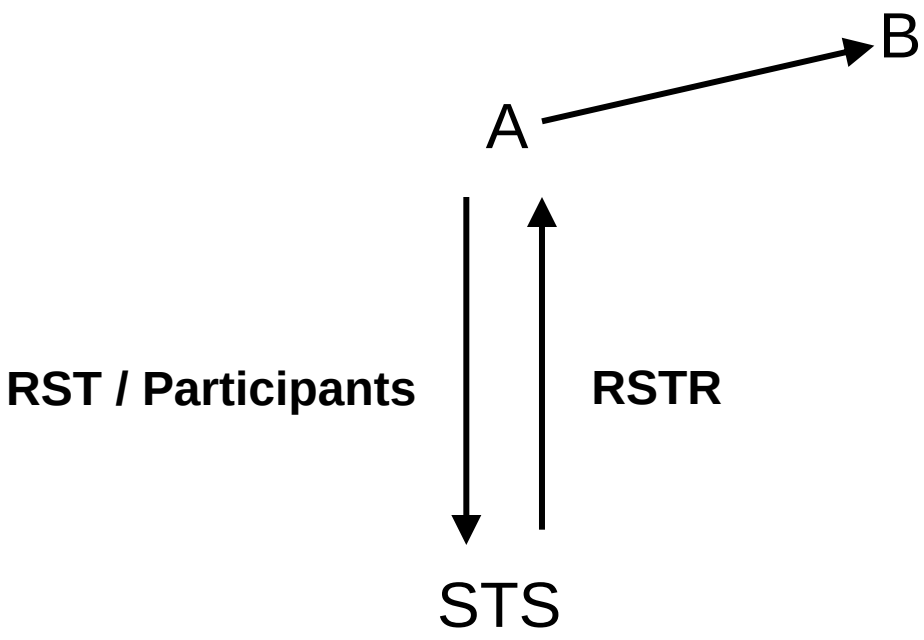
Struktur RSTR



Im Auftrag von / Delegation

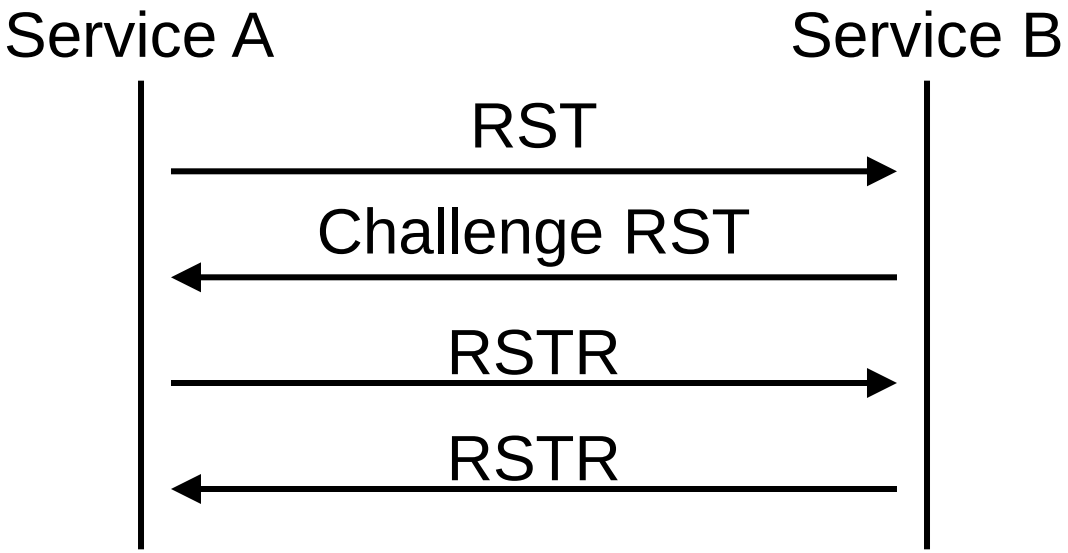


Teilen von Tokens



Challenge Extension

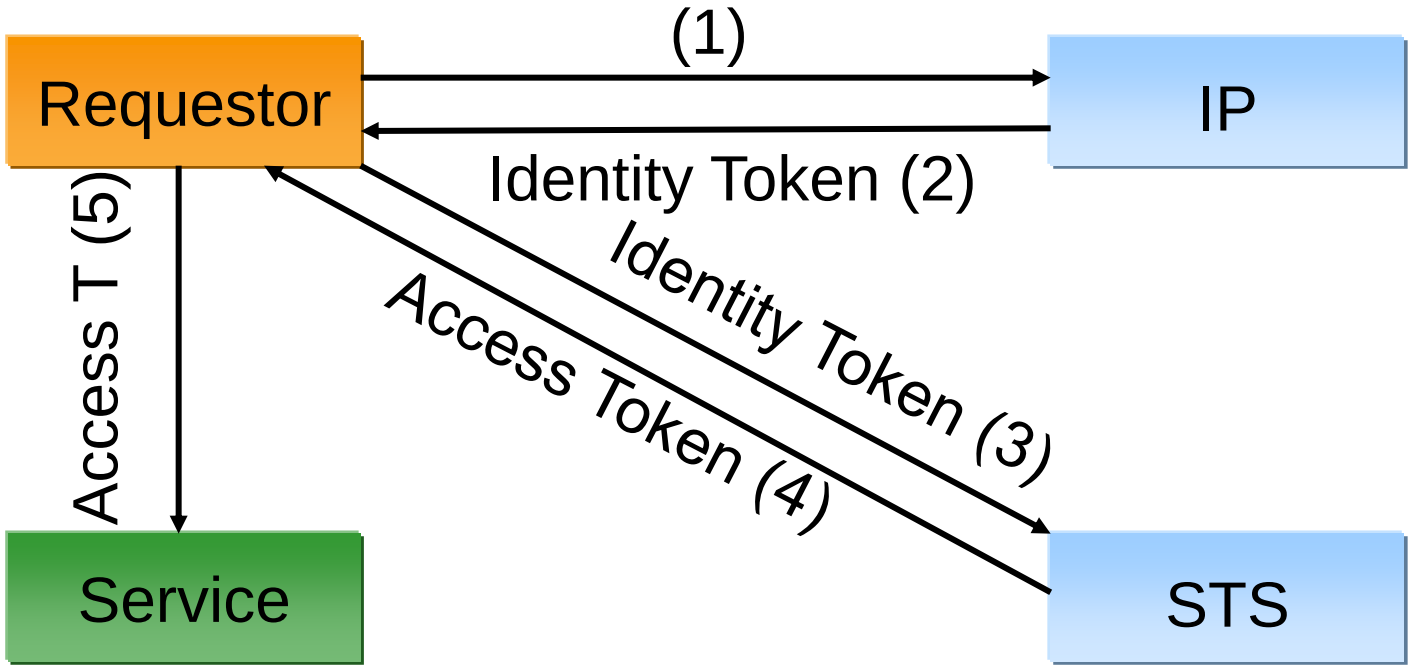
- Requestor wird mit Challenge auf Probe gestellt



WS-Trust Bindings

- Issuance
  - Ausgabe neuer Tokens
- Renewal
  - Erneuern von abgelaufenen Tokens
- Cancel
  - Stornierung eines Tokens
- Validation
  - Validierung von Token

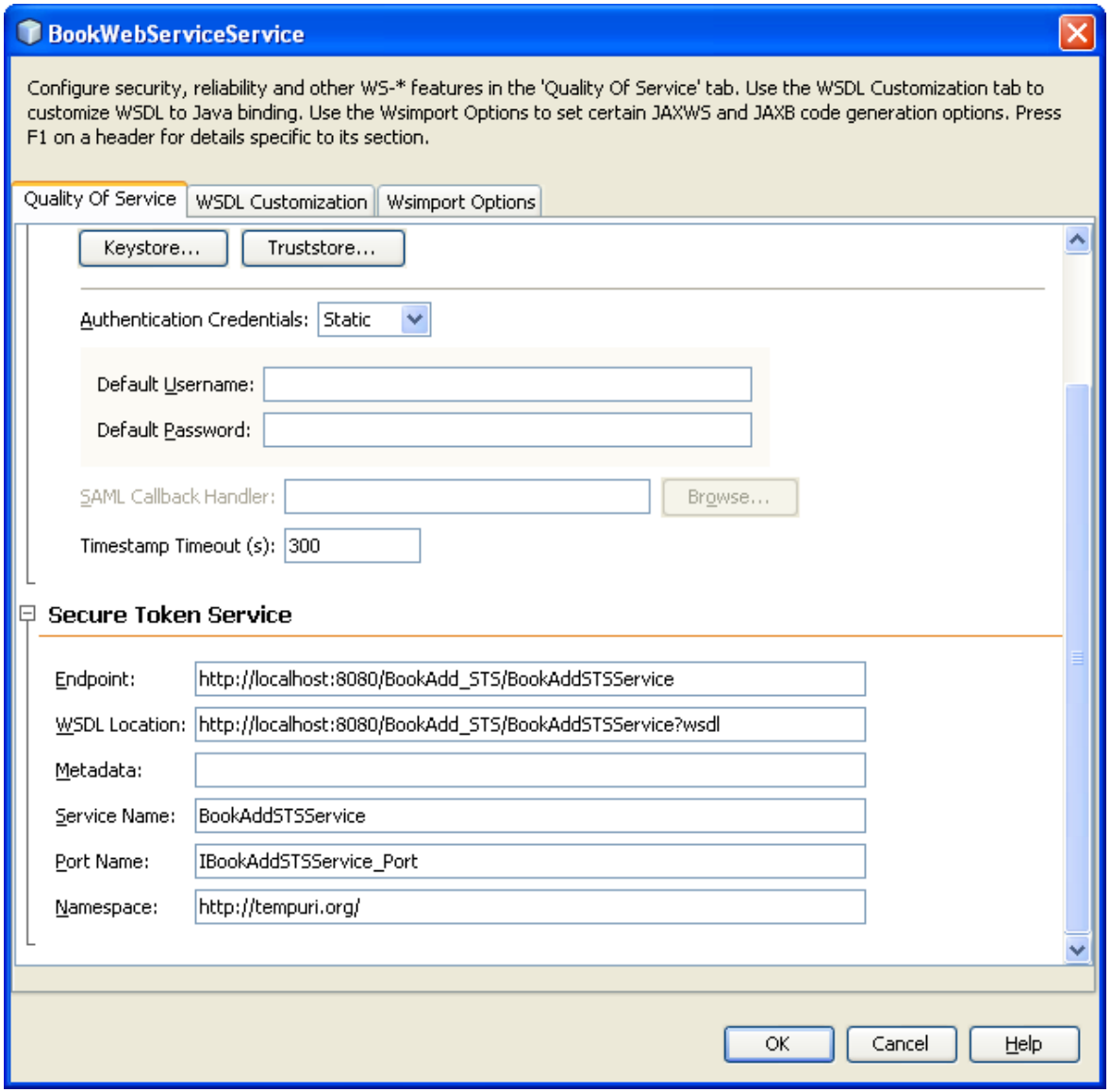
Arbeitsteilung zwischen IP und STS



Fragen

- Welche Token erwartet Provider?
- Welchem STS vertraut Provider?

STS Client Konfiguration in Netbeans



WS-Secure Conversation

WS-Secure Conversation

- Problem: WSS für einzelne Nachrichten
- Etabliert Context über Token
- Sitzungsbasierte Sicherheit durch
  - Session Key
- Definiert Ausgabe von Sicherheitskontext Tokens über WS-Trust
- Oasis Standard Version 1.3 2007
- Nachbau SSL Handshake
- Framework für den Austausch von Token
  - Den Aufbau eines Security Contexts
  - Den Austausch und die Ableitung von Schlüsseln
- Art SSL auf SOAP Ebene

Security Context

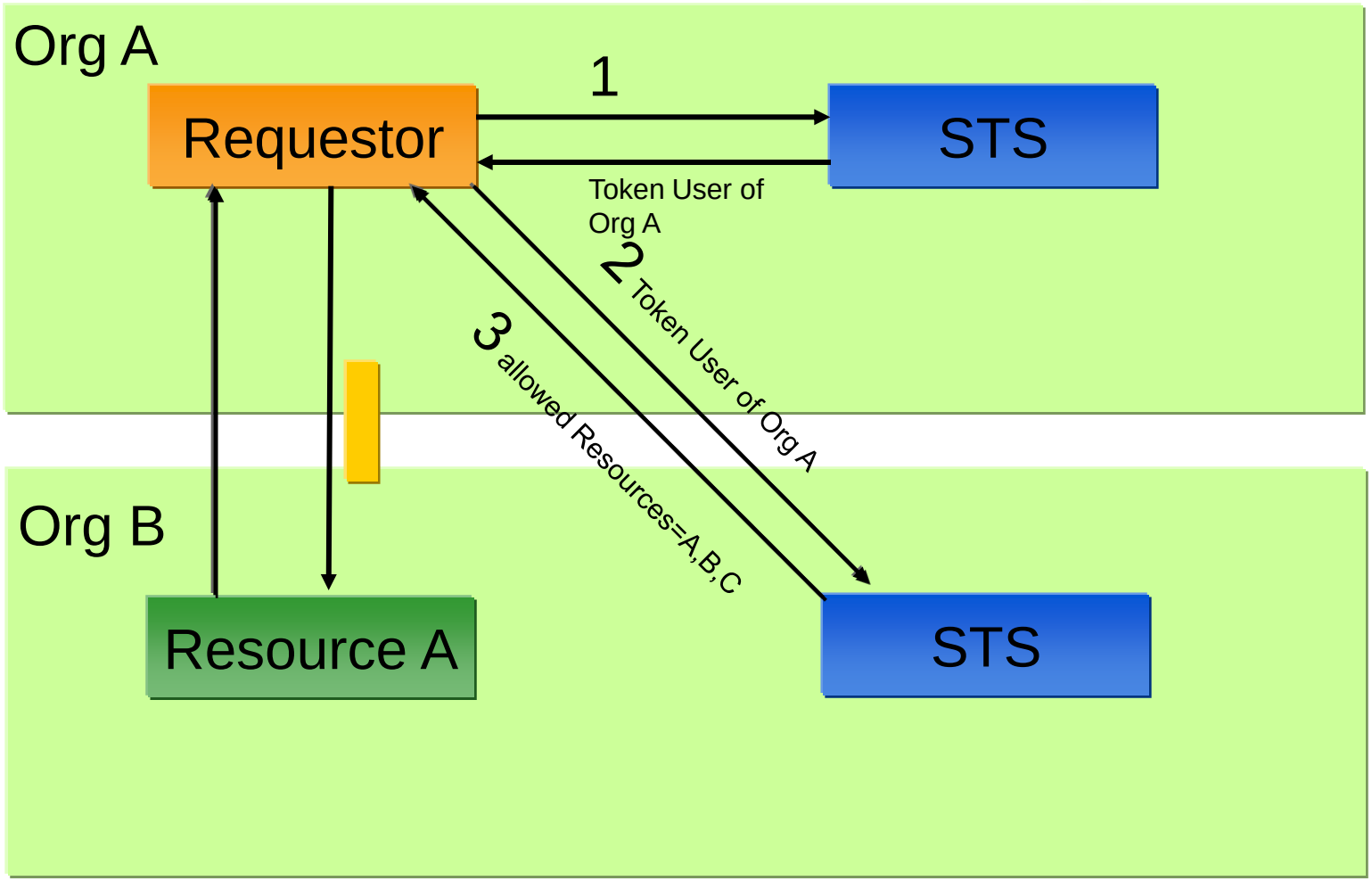
- Ermöglicht den gesicherten Austausch von mehreren Nachrichten
- Lifecycle umfasst gesamte Kommunikation
- Vermeidet Overhead bei der Sicherung der einzelnen Nachrichten

WS-Federation

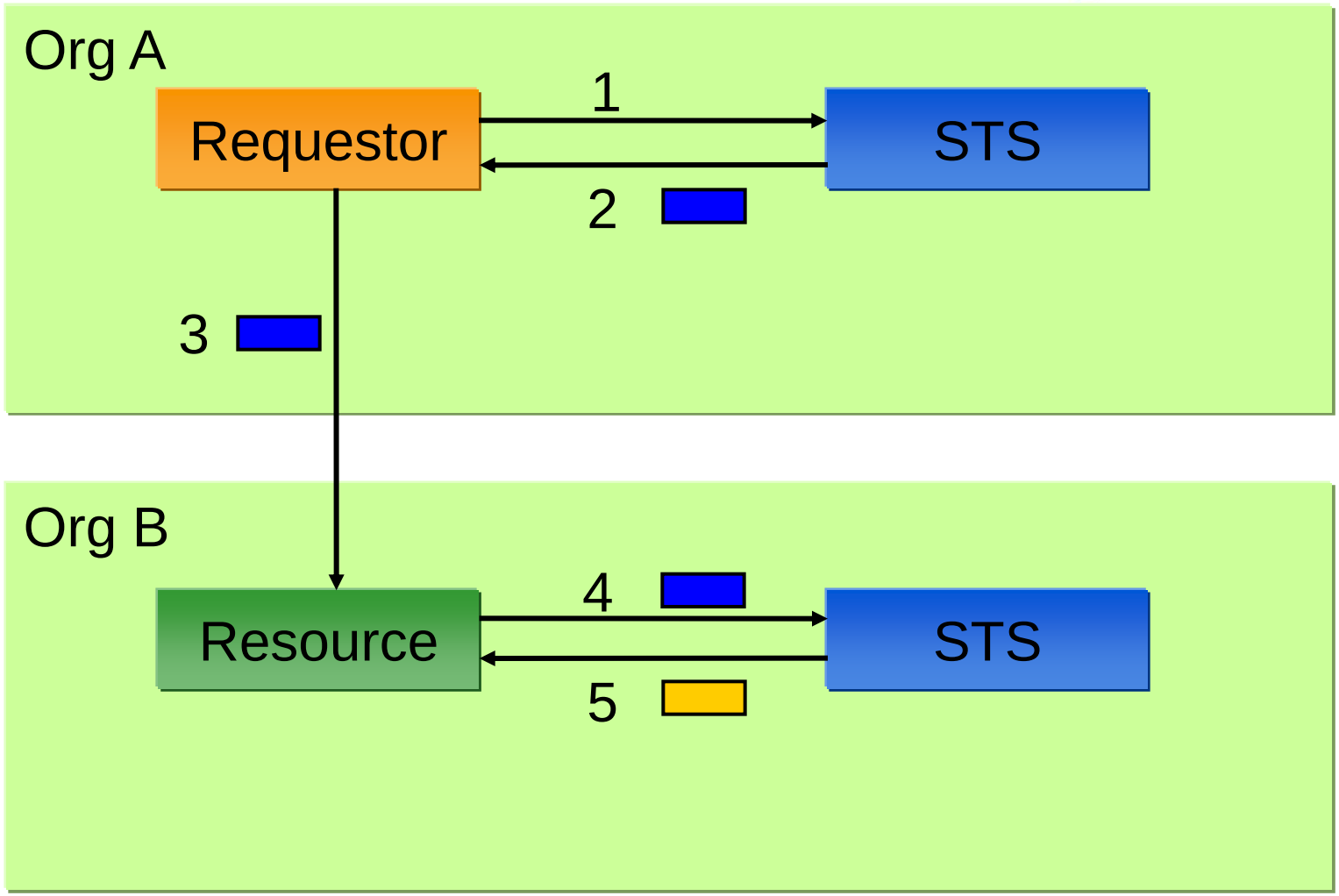
WS-Federation

- Eine Anmeldung -> Zugriff auf Ressourcen mehrerer Organisation
- Keine Zusammenarbeit über eMail
  - „Schick mir mal…“
- Föderation von Realms durch Vermittlung von Vertrauen, Attributen und Authentication
- In Windows Server ab 2005R2
  - Active Directory Federation Service ADFS
- Erweitert WS-Trust
- Unterstützt Web Anwendungen und Web Services
- Bietet Pseudonyme, Attribute, SSO, Single-SignOff
- Version 1.1 wurde im Mai 2007 an Oasis übergeben

Federation Szenario 1



Federation Szenario 2



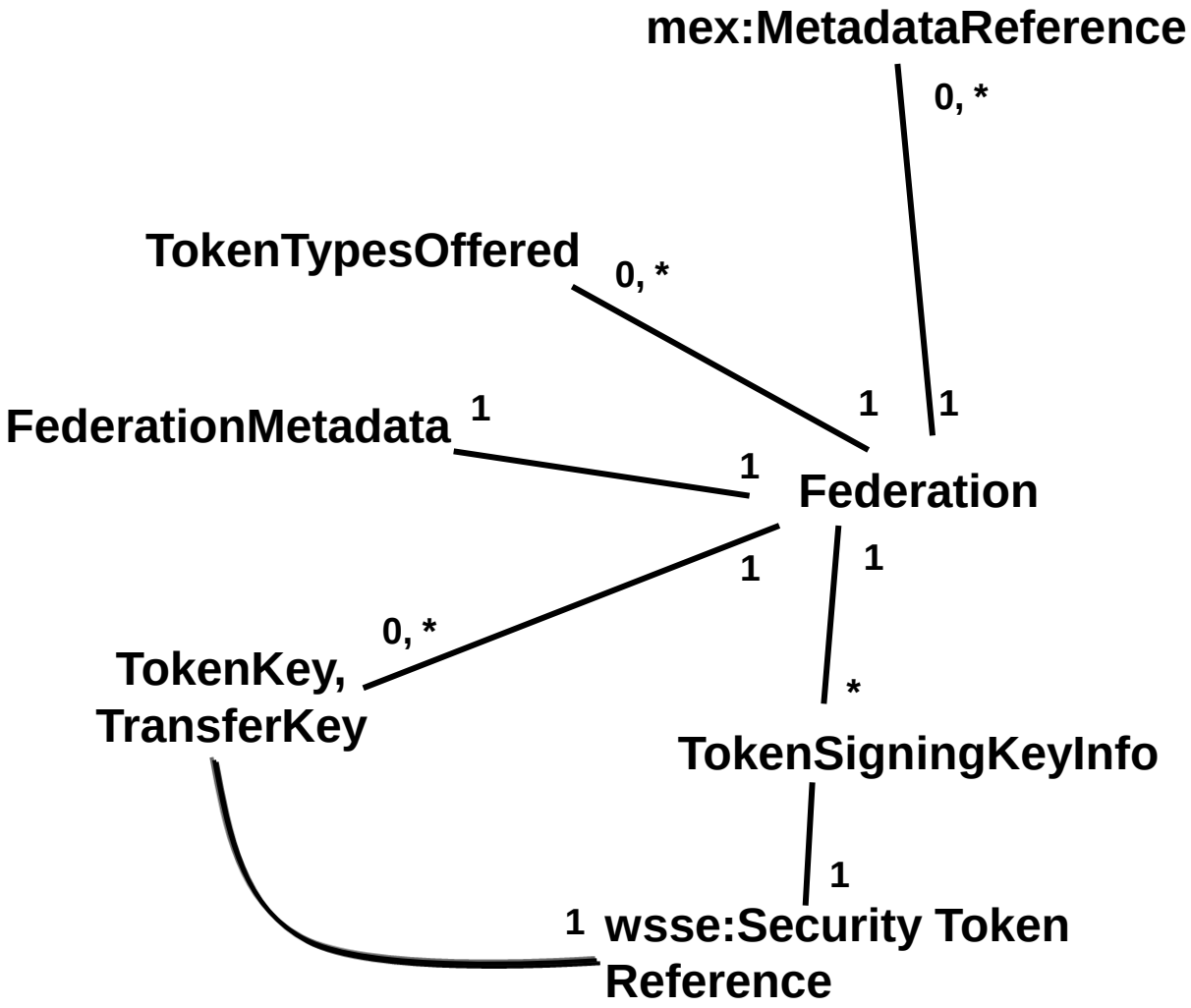
Federation Metadata

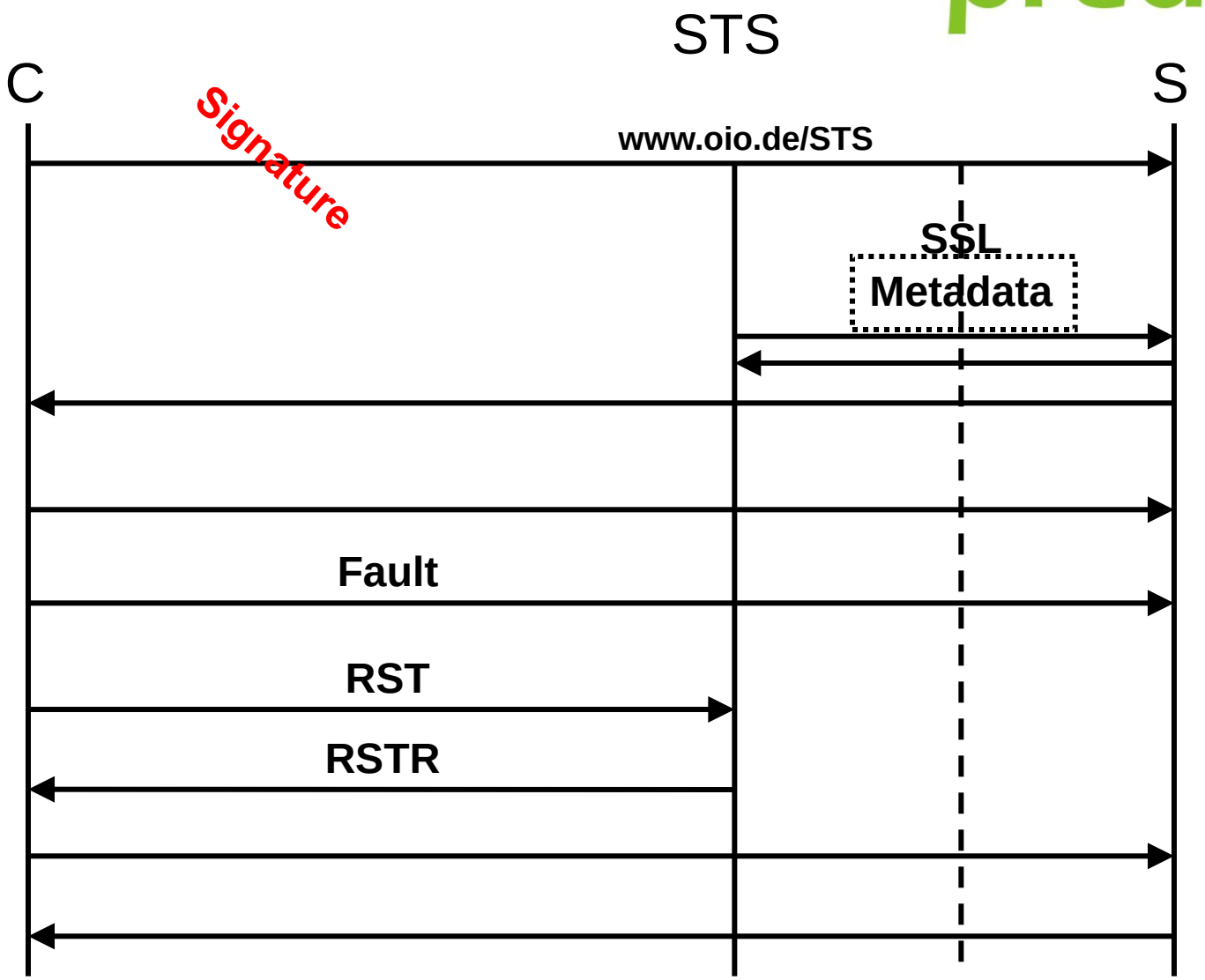
- WS-Federation definiert
  - Metadata Model
  - Dokument Format
- Informationen über beteiligte STS
  - Public Key
  - URL
  - Domains
- Austausch manuell, automatisch oder spontan

Anfordern von Federation Metadata

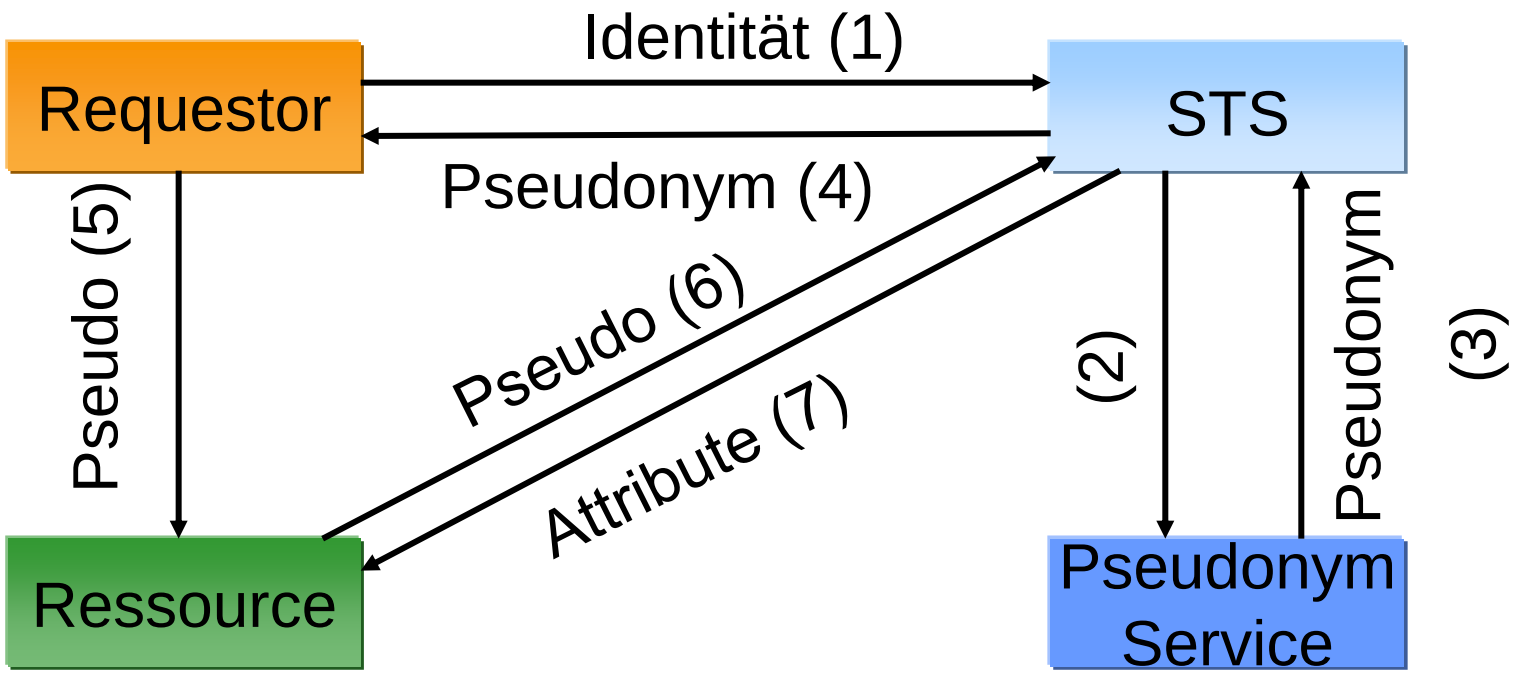
- Bootstrap Reihenfolge
  - 1. Bekannte HTTPS Adressen
  - 2. Well-Knwon HTTPS Adressen
  - 3. Bekannte HTTP Adresse
  - 4. Well Known HTTP Adrese
  - 5. DNS SRV Eintrag
- Download Reihenfolge:
  - 1. HTTPS
  - 2. WS-Transfer/WS-Ressource Transfer
  - 3. HTPP
- Default Metadatapath:
  - https://server/FederationMetadata/spec-version/FederationMetadata.xml

Federation Metadata

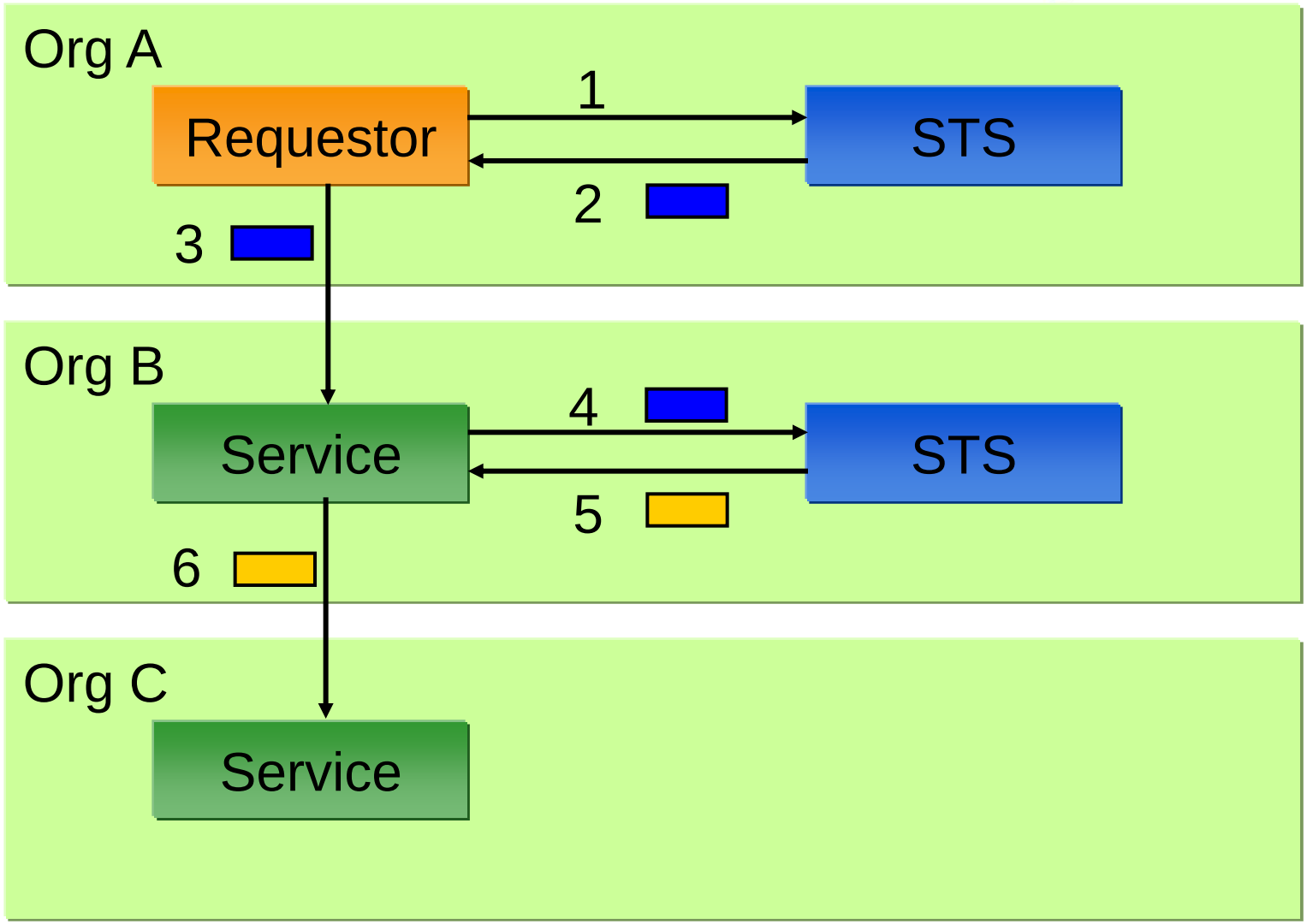




Anonymität über Pseudonym Service



Composite Services



Erweiterung für WS Trust

- Reference Token
  - Kann wie Token benutzt werden
  - Ziel: Effizienz
- FederationID für RST
  - Für welche Federation wird ID benötigt
- Anfrage von Proof Tokens
  - Use Case: Receipient kann Session Key nicht extrahieren
- Client basierte Pseudonyme
  - Client äußert beim STS wunsch über Pseudonym
- Freshness for Tokens

Single Sign On für das World Wide Web

Windows Card Space

- Client für Identity Metasystem
- Basiert auf WS-* Standards

OpenID

- SSO für das Web
- Frei und einfach
- Es gibt mehrere OpenID Provider
- Dezentral
- Benutzt Internet Technologie: HTTP, SSL, URI, liberele Lizenz…
  - Gehört niemandem

OpenID Foundation

- Wird von Community geleitet
- Unterstützt Open Source
- Übernimmt rechtliche Dinge der Community

Sites mit OpenID Unterstützung

- AOL
- Microsoft
- Novell

Kerberos

Kerberos

- Verteilter Dienst für die Authentifizierung
- Ermöglicht die gegenseitige Authentifizierung von Benutzer und Diensten
- Entwickelt vom MIT
- Freie Implementierung
- Basiert auf dem Schlüsselverteilungsmodell von Needham und Schroeder
- Verwendet symmetrische Kryptographie
- RFC 4556 beschreibt wie eine PKI für die Authentifizierungsphase verwendet werden kann.

Bones

- Encryption is a weapon
- Kerberos without calls to DES routines
- John Kohl wrote “piranha” to delete the calls. Piranha eats the flesh- the bones remain.
- Errol Young put encryption into the bones→ E-Bones
  - E=Encrypted

ID Example

- Identity consists of name, address
- Additional information
  - Restrictions, e.g. weight limits for freight
  - Expiration date

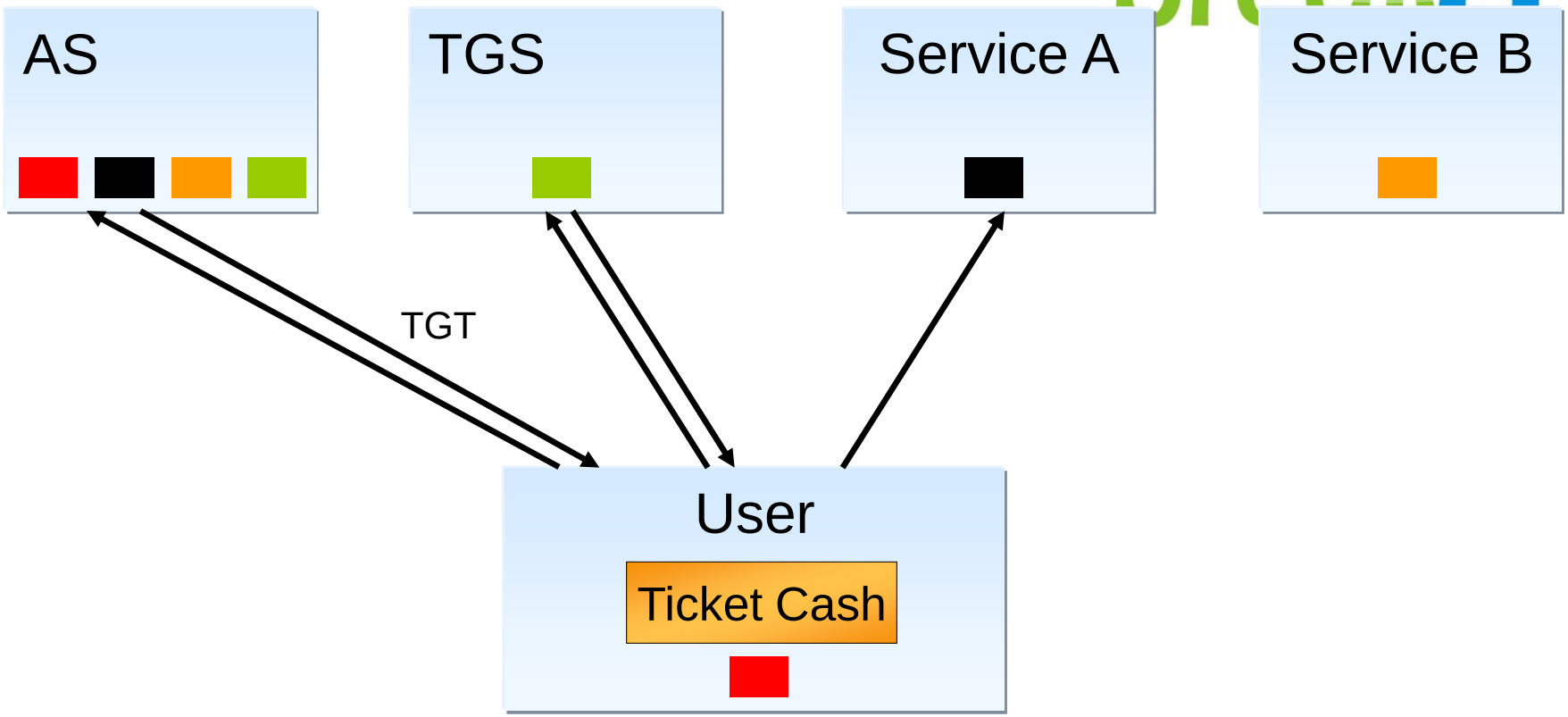
Kerberos Schlüssel

- Kerberos basiert auf gemeinsamen Schlüsseln
- Schlüssel für Verschlüsselung und Entschlüsselung sind identisch
- Teilnehmer beweisen, dass sie den Schlüssel kennen ohne ihn über das Netzwerk zu senden.

Kerberos Authentication Server (AS)

- Vergibt Tickets an Benutzer
- Authentifizierung erfolgt über Passwort
- Benutzer und Dienste hinterlegen Schlüssel beim AS
- Benutzerschlüssel wird vom Passwort abgeleitet
- Schlüssel für Dienste sind zufallsbasiert
- Befreit andere Services von der Verwaltung einer Datenbank mit Benutzerkonten

Geheimnisse in Kerberos



- Geheimnis zwischen AS und User
- Geheimnis zwischen AS und TGS
- Geheimnis zwischen AS und Service A
- Geheimnis zwischen AS und Service B

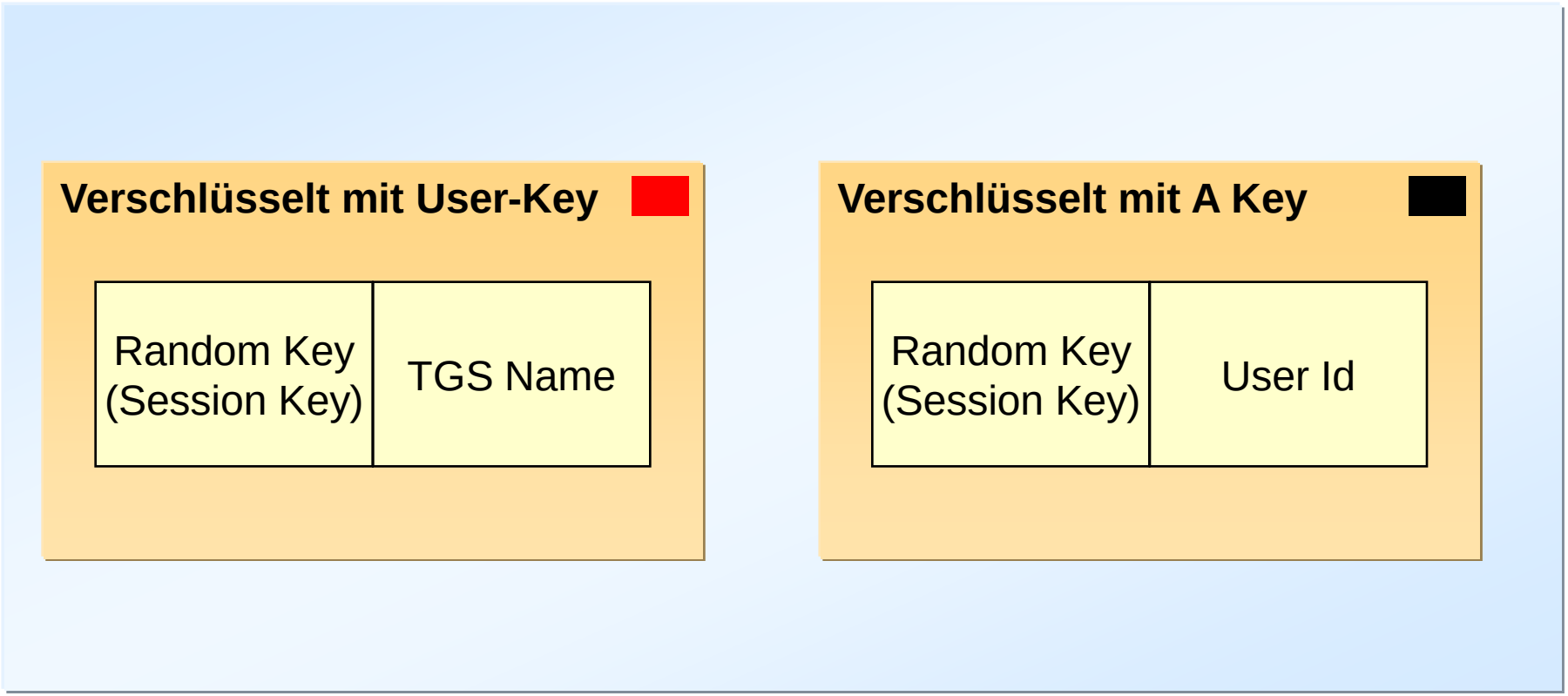
Ticket Granting Server

- AS used every time a service is used
  - ➔ High load on AS
- Division of labor between AS and TGS
- Key Distribution Center
  - AS + TGS
- AS issues ticket granting ticket (TGT) to access TGS
- TGS issues ticket for services

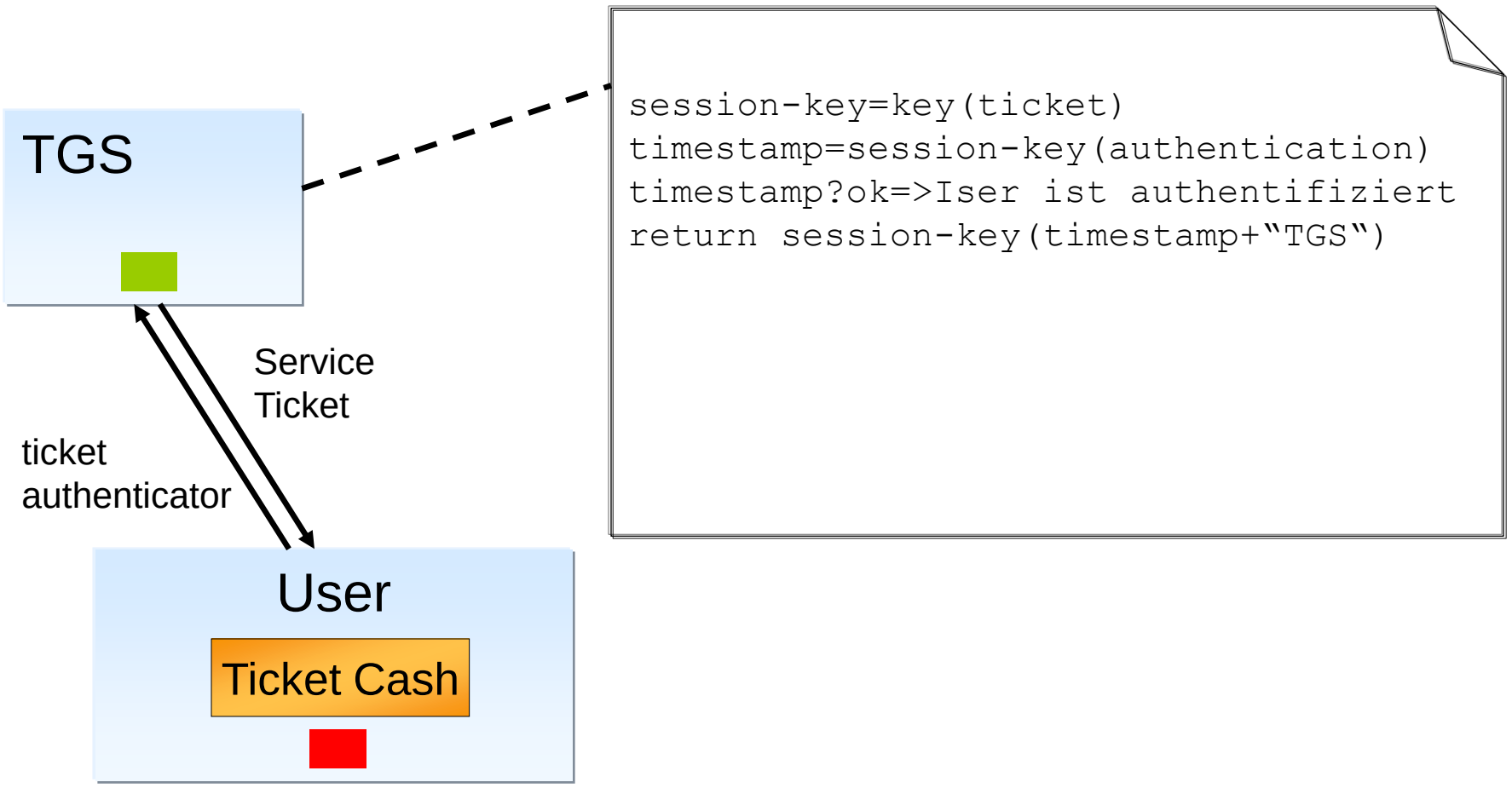
Ticket Granting Ticket

- Wird vom Ticket Granting Service ausgestellt
- Ist nur begrenzt gültig (* 8 Stunden)
- Wird zusammen mit Service Tickets in „Ticket Cache“ oder „Credential Cache“ des Users gespeichert

Ticket Granting Ticket



Anfordern von Service Tickets



Verschlüsselt mit Session Key

Timestamp

Credential Cache

- Tickets typically expire after 8 hours
- Users cache tickets

Resources

- The Moron’s Guide to Kerberos
  - http://www.isi.edu/gost/brian/security/kerberos.html

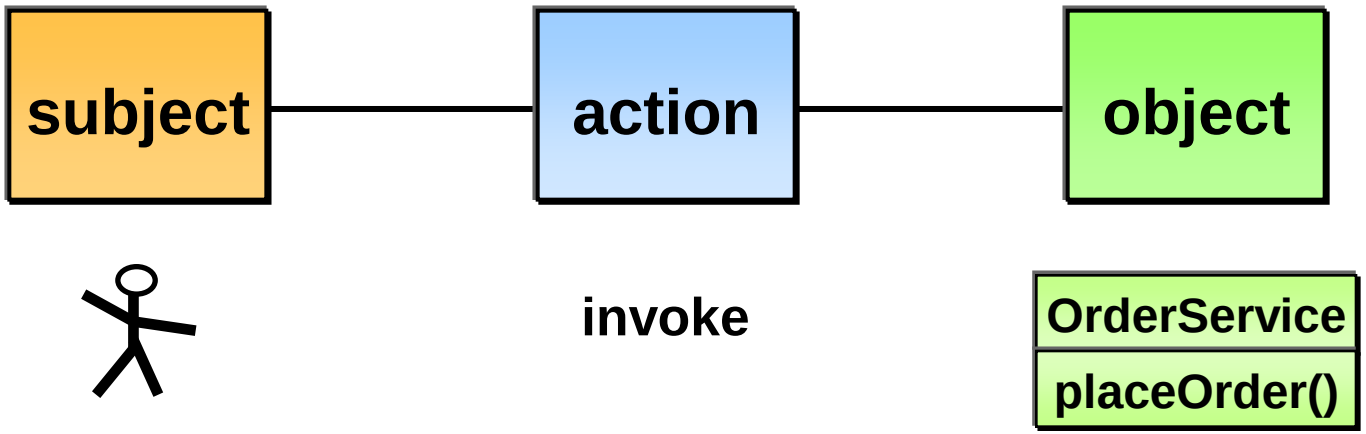
Authorisierung

Security Policy Management

- Policies erstellen
- Überprüfen, Testen
- Freigeben
- Ausgeben
- Kombinieren
- Analysieren
- Modifizieren
- Stornieren
- „Enforcement“

XML Access Control Language XACML

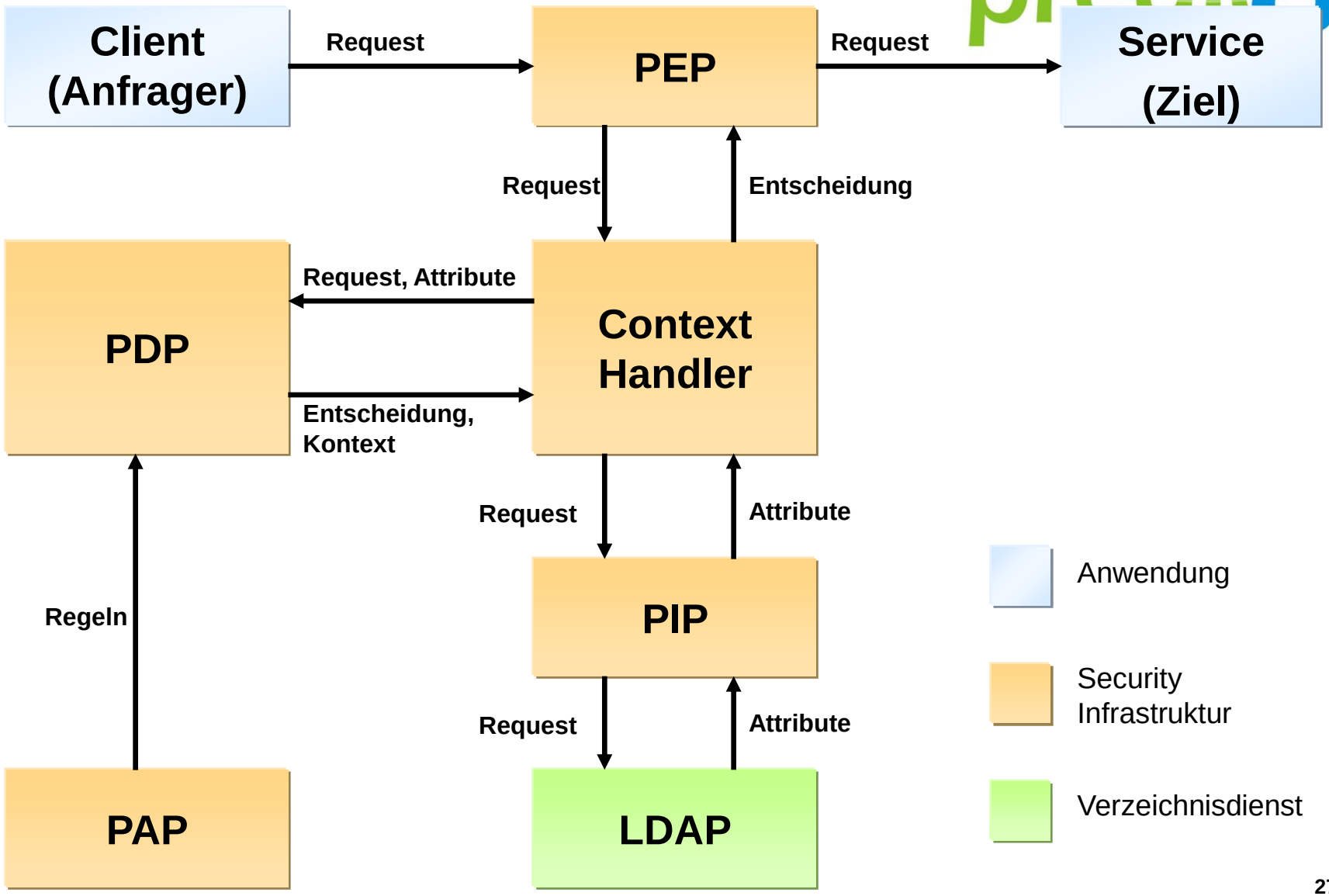
- Ziel: Unternehmensweite Sprache für Zugriffsregeln
- OASIS Standard
- Nicht auf WS begrenzt



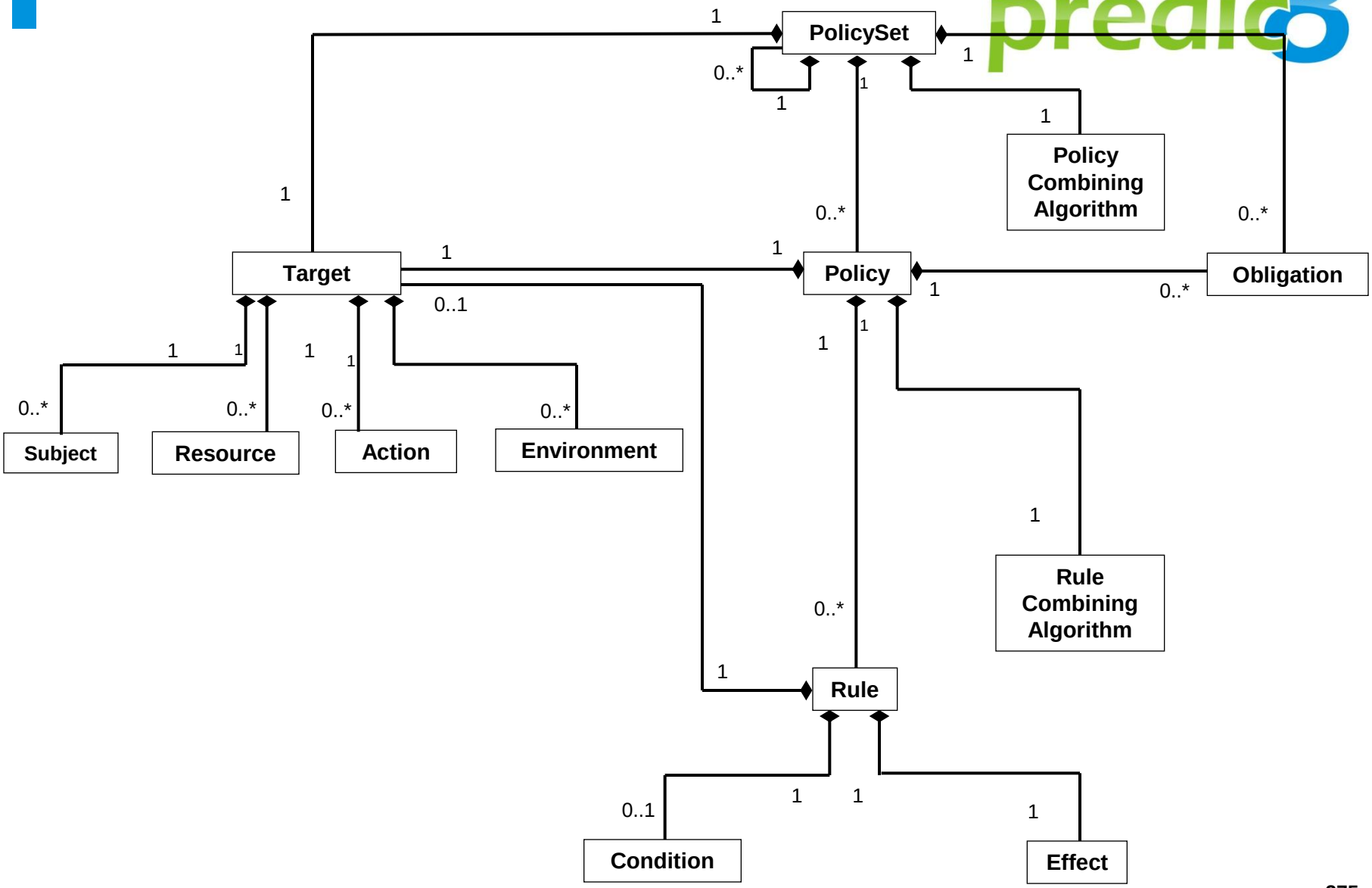
Access Control Glossary

Access Control	Zugriffskontroller auf der Basis einer Policy
Action	Operation auf einer Resource
Authorization Decision	Ergebnis der Auswertung einer Policy “Permit“, “Deny“, “Nichtentscheidbar“
Policy	Menge von Regeln
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PAP	Policy Administration Point
PIP	Policy Information Point

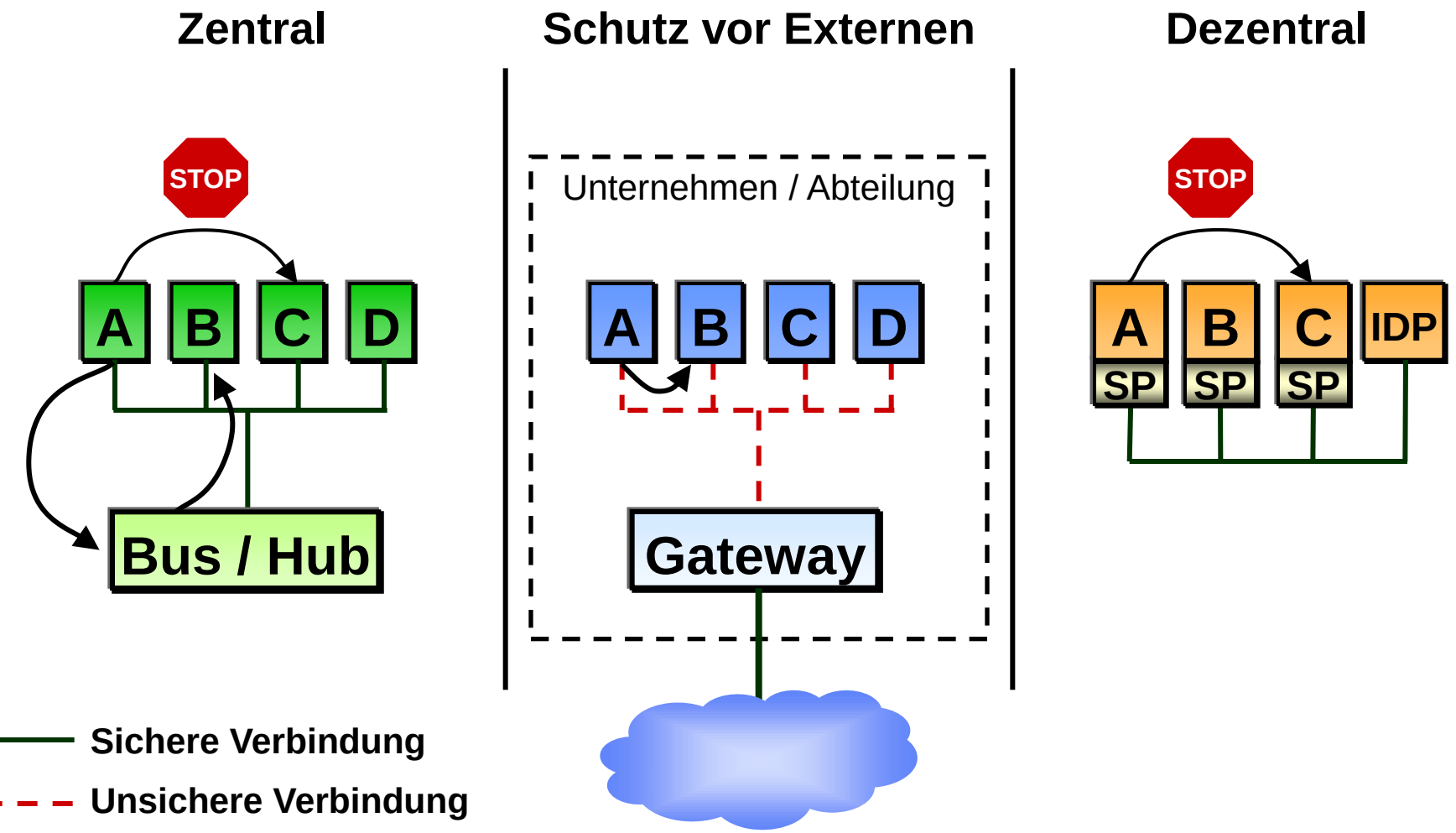
Infrastruktur für Authorisierung



Datenmodell XACML



Topologien



Glossar

EE	EndEntity
PFX	Vorfahre von PKCS #12
ASN.1	Abstract Syntax Notation One
DER	Distinguished Encoding Rules (DER)
XER	XML Encoding Rules
CMP	Certificate Management Protocol
OCSP	Online Certificate Status Protocol
PKI	Public Key Infrastructure
PMI	Privilege Management Infrastructure
POP	Proof of Possession

Quellen

- „Applied Cryptography, Second Edition“, Bruce Schneier, Wiley 1996
- Java Security, http://java.sun.com/security/
- „Kuckucksei“, Clifford Stoll
- „Hacker für Moskau“, Thomas Amman, Matthias Lehnhardt, Gerd Meißer, Wunderlich Verlag (August 1992)