Network Architectures and Services, Georg Carle
Faculty of Informatics
Technische Universität München, Germany

Network Security

Chapter 10

Application Layer Security: Web Services

Technische Universität München

Part I: Introduction to Web Services

- Part I: Introduction to XML and Web Services
- Part II: Securing Web Services
- Part III: Identity Federation

Recap: Internet Protocol Suite

Application Layer	Application protocols: e. g. HTTP, SIP, **Web Services**
Transport Layer	End-to-end connectivity between processes (port concept)
Network Layer	Routing between networks
Data Link Layer	Interface to physical media
Physical Layer	

Web Services: loose definition

- No consensus on a precise definition "in the community"
- **Loose definition: a collection of technologies that employ HTTP technology and enable application interoperation over the Internet.**
- Examples:
  - Web APIs (e. g. Google Maps, …)
    → often used for **mashups**: Web application that combines data from different sources
  - XML-driven Web Services using a variety of XML-based protocols like SAML, WSDL, UDDI
    → often used for Service-Oriented Architectures
  - RESTful services (recent development, out of scope)
- The distinguishing trademarks seem to be:
  - Use of HTTP
  - Application interoperation – not human users

Web Services: Contributors

- We will mostly (but not exclusively) focus on the latter: XML-driven technologies.
- These technologies have been defined in a (large) number of standards and by several committees
- Standardization Committees:
  - OASIS: Organization for the Advancement of Structured Information Standards
    - Large number of members of different membership classes, including many global players like IBM, Microsoft, SUN
    - Responsible for, e.g., many WS-* standards, SAML, UDDI
    - Offers an Identity Federation standard
  - W3C: World Wide Web Consortium
    - Defines WWW standards: “W3C Recommendations”
    - Responsible for, e.g., HTML, XML, XSL-*, SOAP, WSDL
  - Liberty Alliance
    - Offers an Identity Federation Standard
- Many standards were first developed by companies and then brought to the attention of a standardization committee.

Recap: XML

- XML = Extensible Markup Language
- A generic “meta-language”, designed as a set of syntax rules to encode documents. Ideas:
  - Separate document content from its representation
  - Machine-readable, but accessible for humans
  - XML is practically a subset of SGML (Standard Generalized Markup Language) from the 1980s
- Representation rules are stored in different documents → allows to define different representations for all kinds of output formats (HTML browsers, PDF, audio…)
- XML is used to define many markup languages you know:
  - HTML → called XHTML
  - XSL-T: transformation into other (markup) languages
  - XML Schema: used to define a markup language (!)
- Many related standards:
  - XPath: access parts of documents
  - XSL-FO: representation for a rendering device, e.g. PDF renderer (ironically, defined in prose: XML Schema is not powerful enough…)
- XML is used in the definition of practically all Web Services Standards!
  - E.g. SOAP, WSDL

Example: XML Fragment

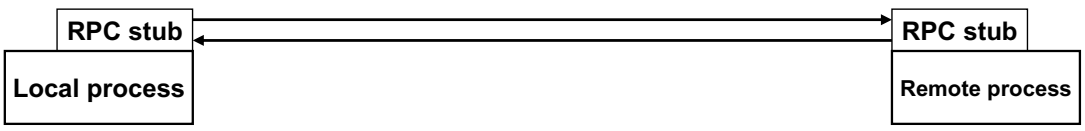
- XML documents can be **highly complex**:
  - Tree structured
  - Can be deeply nested
  - White spaces etc. have no semantics → only structure counts
- This makes **parsing XML computationally very intensive**
- Example (a kind of policy document that we wrote once):

```
<accessControl>
      <policy id="0">
         <requirement>
            <dataItem name="CERTIFICATE_VERIFICATION" />
         </requirement>
         <control>
            <action id="VERIFY_CERT" retType="BOOL"/>
            <argument position="4" type="LIST">
               <item>7831fd24756d1c645843c9016c6bae2d20f476bc</item>
               <item>ff3fa9d8c509b254c3fa185bd4b85b3c9eb84a3d</item>
               <item>6dbd654eae4bd8b8a1151d8b4b5d197a275efdf3</item>
            </argument>
         </control>
         <nextStateOnPositiveEvaluation id="1"/>
         <nextStateOnNegativeEvaluation id="NACK"/>
      </policy>
</accessControl>
```

Important concepts:
- **Element ordering does not matter!**
- **White spaces do not matter!**

Web Services as a Middleware

- In many respects, **Web Services are similar to Remote Procedure Calls (RPC)**
  - Used like a local function:
    - Parameter marshalling
    - Call to remote process with parameters
    - result is returned by remote process
  - Middleware can abstract over the particularities of the communication over the network
  - Loose coupling (asynchronous)
  - Web Services are generally more complex than a simple RPC
  - But there is also a standard for RPC: XML-RPC ☺
- Web Services are **realized with with HTTP and XML**



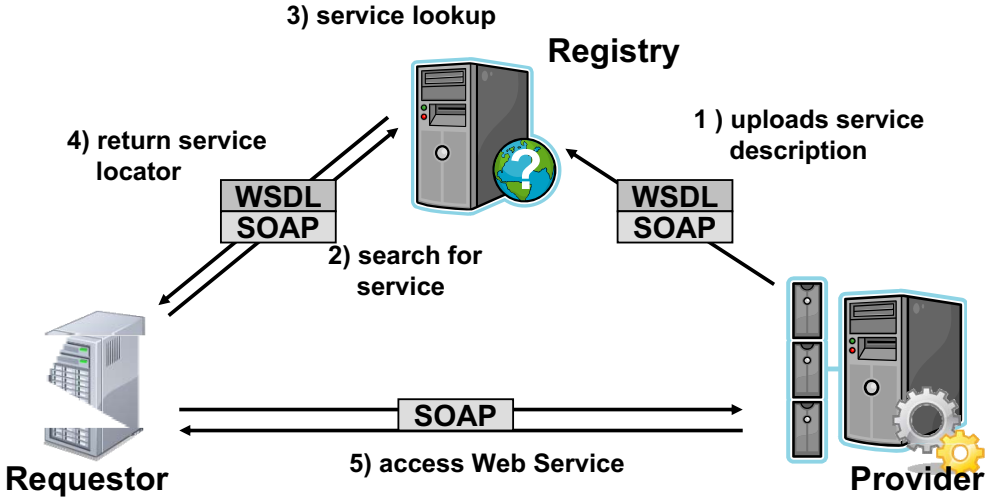
Simplified RPC example

Web Services as a Middleware

- **Why HTTP?**
  - Because HTTP technology is around, well-supported and well-accepted → easy to win support
  - But state-less property is not in favour of HTTP as Web Services realize complex work flows
- **Why XML?**
  - Already well-accepted: easy to win support from vendors
  - XML is often already a company-internal format: no conversion necessary
  - Easy to define your own, domain-specific language (e. g. B2B)
  - Relatively easy to define service composition and orchestration
- **Why not XML?**
  - Parsing very slow
  - White spaces and element ordering: have impact on encryption and signing
- On the whole, the advantages of HTTP/XML outweighed the drawbacks
  → Web Services are, after all, a very industry-relevant concept

Original Vision of Web Services

- **Building blocks: a classic architecture**
  - **Service description**
    WSDL: Web Services Description Language
  - **Service discovery: via a registry**
    (possibly UDDI = Universal Description, Discovery and Integration)
  - **Interaction: carrier protocol: SOAP**
    (SOAP used to be an acronym, now it's just a name)



More Web Services Visions

- **Service composition:**
  - A service is composed of many sub-services, all defined in WSDL
- **Business-process orchestration:**
  - Model business processes with Web Services
  - WS-BPEL: Business Process Execution Language
- **Service-Oriented Architecture:**
  - Paradigm of software architecture
  - Realize functionality as a composition of services
  - Can be done with Web Services (other frameworks possible)
- We will not go into more detail here
- The idea you should get is: **Web Services can be very complex**
- **Complexity means attack surface and room for mistakes**
  … especially if:
  - Self-designed interactions are possible
  - Self-designed security mechanisms are possible

Web Services: Our Focus

- There are a **large** number of Web Service standards
- Many complement each other, some are competitors
  - Just browse oasis-open.org
- We will mostly discuss:
  - SOAP + XML Encryption + XML Signature
  - SAML: Security Assertion Markup Language
  - Identity Federation standards

- It is almost impossible to discuss all aspects of Web Services and their security
  - There are whole lectures just on this topic (use Google)

- *“The nice thing about standards is that there are so many to choose from. And if you really don't like all the standards you just have to wait another year until the one arises you are looking for.”*
  - A. Tanenbaum

Part II: Securing Web Services

- Part I: Introduction to XML and Web Services
- Part II: Securing Web Services
- Part III: Identity Federation

Securing Web Services

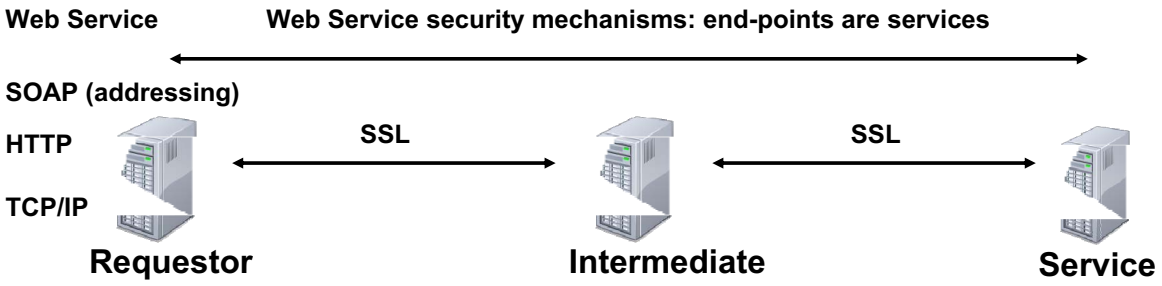
- **Security Challenges**
  - Securing Identities
  - Securing Messages
    (Web Service communication is always message-based)
  - Securing multi-hop message flows
    - In particular important for Service Oriented Architectures
- **Web Service security and other protocols** are not mutually exclusive:
  - You can use SSL to create a secure pipe **between hosts**
  - Interoperate well with, e. g.
    - X.509 certificates (if you have a PKI)
    - Kerberos tokens
- **Question: why not just use SSL?**
  - SSL secures an underlying TCP/IP connection (“bit pipe”)
    → SSL is point-to-point between hosts
  - The identity of a Web Service is not the identity of the bit pipe:
    different end-point (service, not host)
  - SSL is no help in multi-hop scenarios

Securing Web Services

- Web Service documents may be business-relevant:
  may need **legally binding signatures**
  → SSL would secure the wrong endpoint
  → Same reason why you encrypt e-mails:
  the endpoint is not the host, but the (human) user

- Web Service documents may pass intermediaries
  (e. g. SOA, service orchestration):
  these may inspect & change documents en route

  → SSL provides only end-to-end semantics
  → Need for cryptographics mechanisms that allow such modifications



Security-relevant Standards

- A **number of standards** address security for Web Services
- **XML Encryption (XML-Enc, W3C)**
  - Defines how to encrypt XML content
- **XML Digital Signature (XML DSig, W3C)**
  - Defines how to sign XML content
- **WS-Security (OASIS)**:
  - Describes how to use XML Encryption and XML Digital Signature to secure SOAP messages
- **SAML: Security Assertion Markup Language (OASIS)**
  - Describes how to create and exchange authentication and authorization tokens (“assertion”)
  - Designed for interaction between an Identity Provider and an Service Provider
  - Uses XML-Enc and XML Dsig

- We will first look at XML Dsig

XML Signature (XML DSig)

- **Idea: sign an XML document**
- **Stated goals**:
  - Message Integrity
  - Origin Authenticity (sender)
  - Non-repudiation
    - This is remarkable as meaningful non-repudiation needs secure time-stamping
- Supports the **usual cryptographic mechanisms**:
  - HMACs (shared key cryptography)
  - Signatures with public-key cryptography
- An **XML signature is an XML fragment** itself
- However, the signature mechanism has been modified to work with **XML's pecularities**

XML Signature

- **Designed to**:
  - **Sign** anything that can be **referenced by a URI**
  - Even if the URI points to a location **outside** of the XML document (!)
  - Thus, can be applied to a part of the document or the whole document, or also some external document
  - Multiple signatures on a document are allowed (business uses, multi-party agreement etc.)
- **Pitfalls**: because you sign XML, you need to take care of:
  - White spaces (tabs vs. spaces)
  - Line endings (Windows world vs. UNIX world…)
  - Character-set encoding (UTF-8? ISO-8859-1?)
  - Escape sequences
  - etc.
- XML Signature thus needs to **canonicalize the document prior to signing**

Canonicalization (C14N) rules

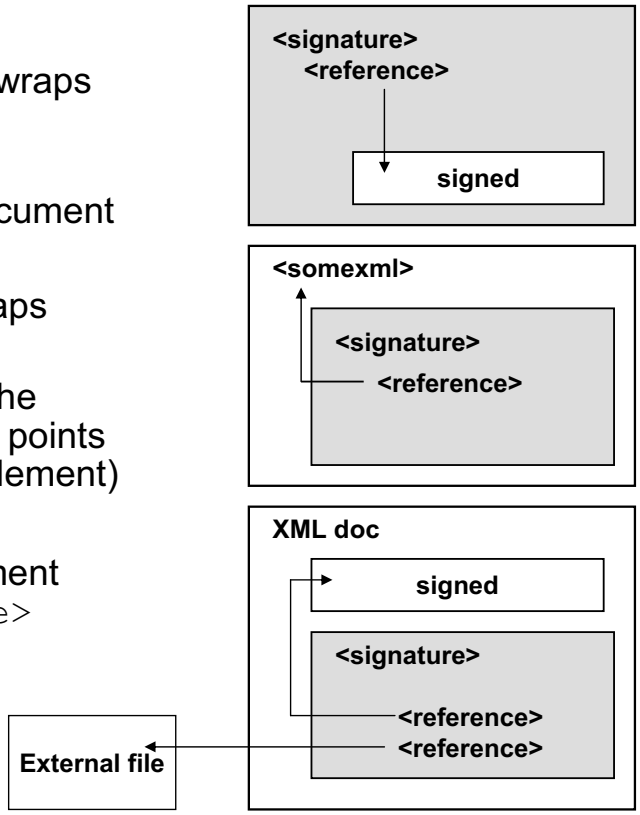
- Steps to take:
  - Encode in UTF-8
  - Normalize line breaks
  - Normalize attribute values
  - Replace character and parsed entities
  - Replace CDATA sections
  - Remove XML declaration and DTD definition
  - Normalize <element /> to <element></element>
  - Normalize whitespaces outside the document and outside elements
  - But retain it within character content
  - Set attribute value delimiters to double quotes
  - Replace special characters in attribute values and character content with character references
  - Remove superfluous namespace declarations
  - Add default attributes to each element
  - Order lexicographically on namespace declarations and attributes of each element
- Question: how fast, do you think, did programmers come up with interoperable implementations? (it took a while)

Transformations

- XML DSig allows to apply five different types of **transformations** when signing
  - Base64 (not a concern)
  - XPath Filtering (selection of elements)
  - Enveloped Signature transform (→ type of signature)
  - XSL-T transform (→ change document tree)
  - Canonicalization
- The transformations are referenced from within the signature
- They need to be reversed before the signature can be validated

Three Kinds of Signature

- **Enveloping signature**
  - `<signature>` element wraps around whole document
  - Signature is stored and referenced inside the document
- **Enveloped signature**
  - `<somexml>` element wraps around document
  - Signature stored inside the document, but reference points to `<somexml>` (parent element)
- **Detached signature**
  - Reference points to element outside the `<signature>` element's hierarchy
  - Can be inside the XML document, or outside



Code Example

- Simplified enveloping signature (not a correct document)

```
<Signature xmlns=http://www.w3.org/2000/09/xmldsig#>
  <SignedInfo>
    <CanonicalizationMethod
      Algorithm=http://www.w3.org/TR/2000...” />
    <SignatureMethod Algorithm=http://www.w3.org...” />
    <Reference URI=“#important”>
      <DigestMethod Algorithm=http://www.w3.org/...” />
      <DigestValue>60nvZ+TB7...</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>ae5fb6fc3e…</SignatureValue>
<KeyInfo>FAE6C...
  <KeyValue>
    <RSAKeyValue>
      <Modulus>uCiu...</Modulus>
    </RSAKeyValue>
  </KeyValue>
</KeyInfo>
 [...]
<Object>
  <SignedItem id=“important”>Secret stuff</SignedItem>
</Object>
</Signature>
```

Discussion of XML Signature

- **Performance:**
  - Signed documents become very large
  - Parsing, canonicalization and transformation are slow
  - Inclusion of external documents allows to design malicious documents that keep referencing more documents (DoS on the parser)
- **Complexity:**
  - Three different kinds of signature
  - Five kinds of transformations that can be applied before signing
  - Complex canonicalisation rules
  - Nonsensical possibilities to signed data and signatures are not explicitly forbidden: signature before signed data etc.
  - Makes analysis of the standard very difficult
- **Correct and comprehensive implementation is difficult**
  → interoperability is easily threatened

Discussion of XML Signature

- If **applied in a sane manner**, the **standard does provide security**
- But you need to be very careful what transformations etc. you allow
- If you want to preserve all XML features, you probably need such a complex mechanism
  → XML Signature is often called a semantic signatures, because it does not just read in the whole document and sign that
- It should be noted that other mechanisms for signatures have been developed:
  - XML-RSig: Really Simple XML Signature:
    "read the BLOB and sign it"
  - XMPP (Jabber): MIME body parts → easier
- Complex overhead is a trade-off usefulness vs. feasibility

Some opinions on XML Signature

- XML Signature has drawn both praise and severe criticism for its flexibility
- Some quotes:
  - "*XML Digital Signature is the latest and greatest technology for you to ensure integrity and non-repudiation. Its remarkable flexibility allows you to sign parts or all of XML documents as well as binary and remote objects.*"

    - J. Rosenberg, D. Remy in [RoRe2004]

  - "*They reinvented the wheel in XML, but made it square to avoid accusations that they'd just reinvented the wheel.*"
  - "*Secure XML, the definitive reference on the topic, spends fully half of its 500-odd pages trying to come to grips with XML and its canonicalisation problems.*"

    - P. Gutmann, University of Auckland [Gu2004]

XML Encryption (XML-Enc)

- Idea: **encrypt XML content**
- **Goal: Confidentiality**
- Just like XML Digital Signature, XML-Enc is agnostic to crypto algorithms
  - You can use it with, e. g.
    - Shared-key cryptography (3DES, AES, …)
    - Public-key cryptography (RSA, …)
  - Usual pattern with public-key cryptography:
    - Generate a symmetric key K
    - Encrypt only K with public key
    - Use K to encrypt the real content
- XML Encryption shares many features with XML Signature

XML Encryption

- Uses `<EncryptedData>` element
  - Either points to encrypted content
  - Or replaces unencrypted content
    (if content is within same document)
- XML-Enc and XML DSig are designed to be used together

Code Example

- Simplified again

```
<MyDoc>
<EncryptedData Id="encdata" xmlns="http://…">
  <EncryptionMethod Algorithm=http://... />
  <CipherData>
    <CipherValue>…</CipherValue>
  </CipherData>
  <EncryptionProperties>
    <EncryptionProperty Target="encdata">
      <EncryptionDate>2010-01-01</EncryptionDate>
    </EncryptionProperty>
  <Object id="encdata">...</Object>
  <Signature>
    …
  </Signature>
</MyDoc>
```

For the last lecture next week…

- We'll probably do some SAML and then Identity Federation
- Maybe OpenID, probably not XACML

- Philosophy round?
  - Fun crypto mechanisms like OTR
  - PKI vs. Webs of Trust
  - P2P Security? → P2PSec lecture…

References

[XMLEnc] W3C. *XML Encryption*. http://www.w3.org/standards/techs/xmlenc.

[XMLDSig] W3C. *XML Signature*. http://www.w3.org/standards/techs/xmlsig

[Gu2004] P. Gutmann. *Why XML Security is Broken.* http://www.cs.auckland.ac.nz/~pgut001/pubs/xmlsec.txt. 2004.

[RoRe2004] J. Rosenberg, D. Remy. *Securing Web Services with WS-Security*. SAMS Publishing. 2004.

[XMPPSig] RFC 3923. *End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)*.

[iSecAttack] iSEC Partners. Attacking XML Security. http://www.isecpartners.com/files/iSEC_HILL_AttackingXMLSecurity_bh07.pdf