Lesson 12 – Introduction to Web Service Security

Service Oriented Architectures Security

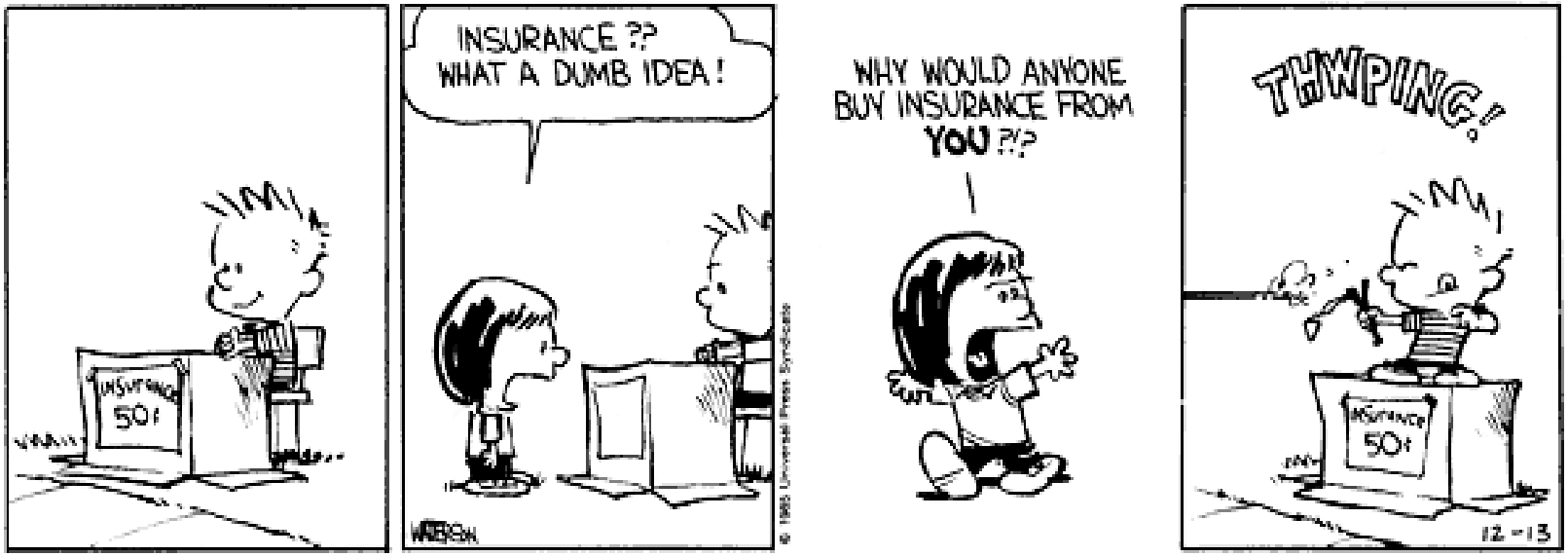
Module 2 - Web Service Security

Unit 1 – Auxiliary Protocols

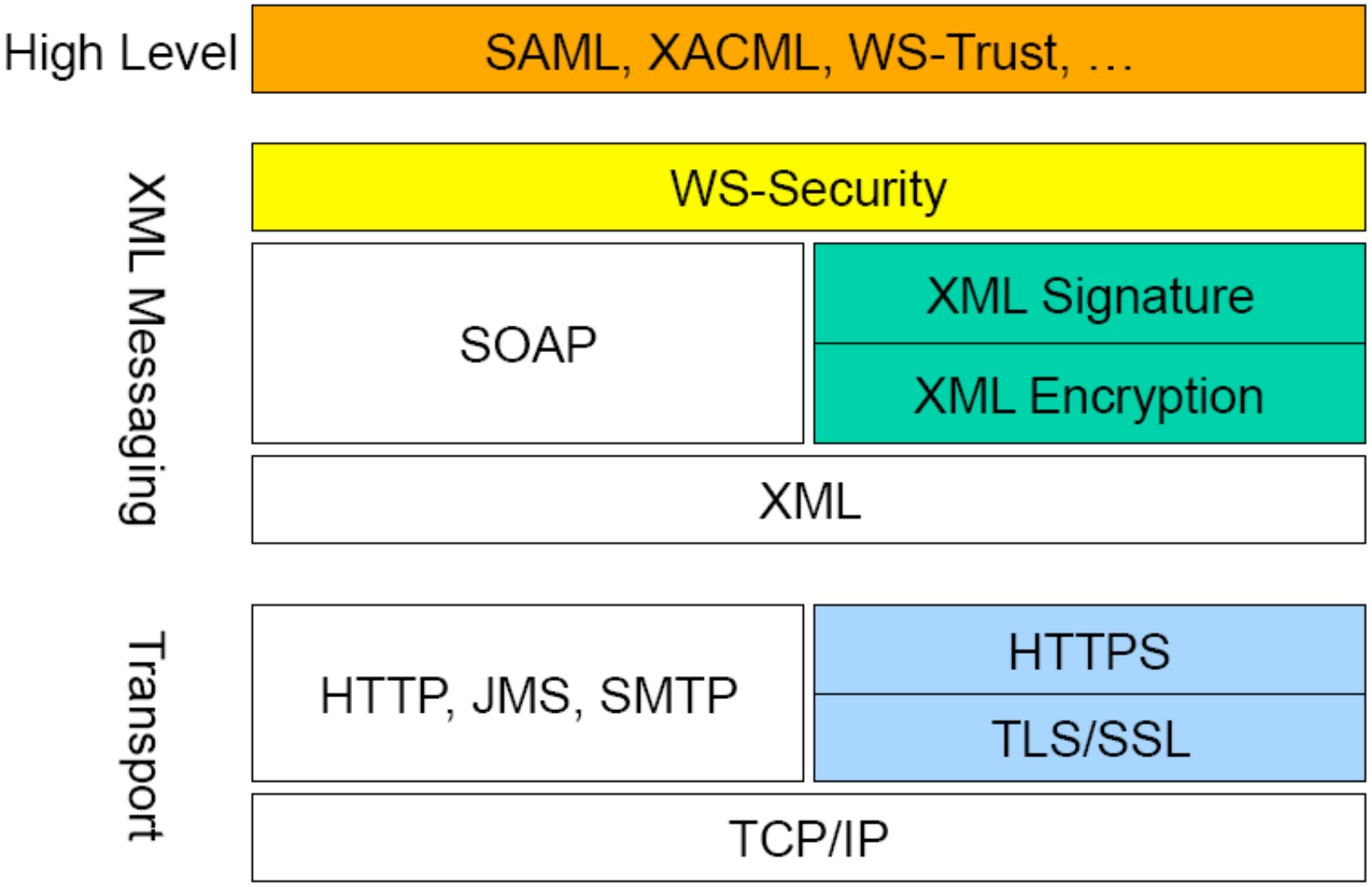
Ernesto Damiani

Università di Milano

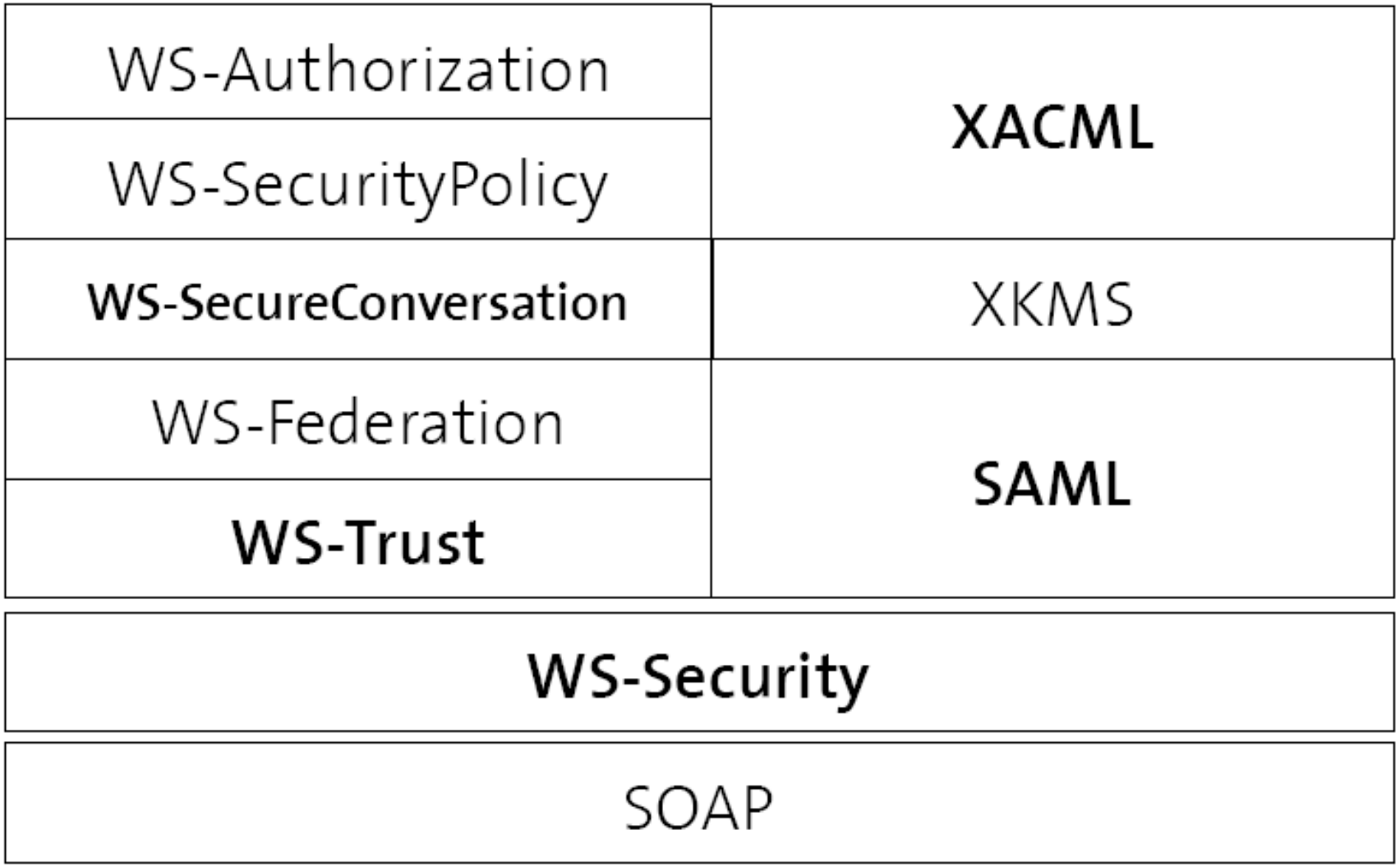
Web Services Security Standards



Security Standards Overview



Security Standards Stack



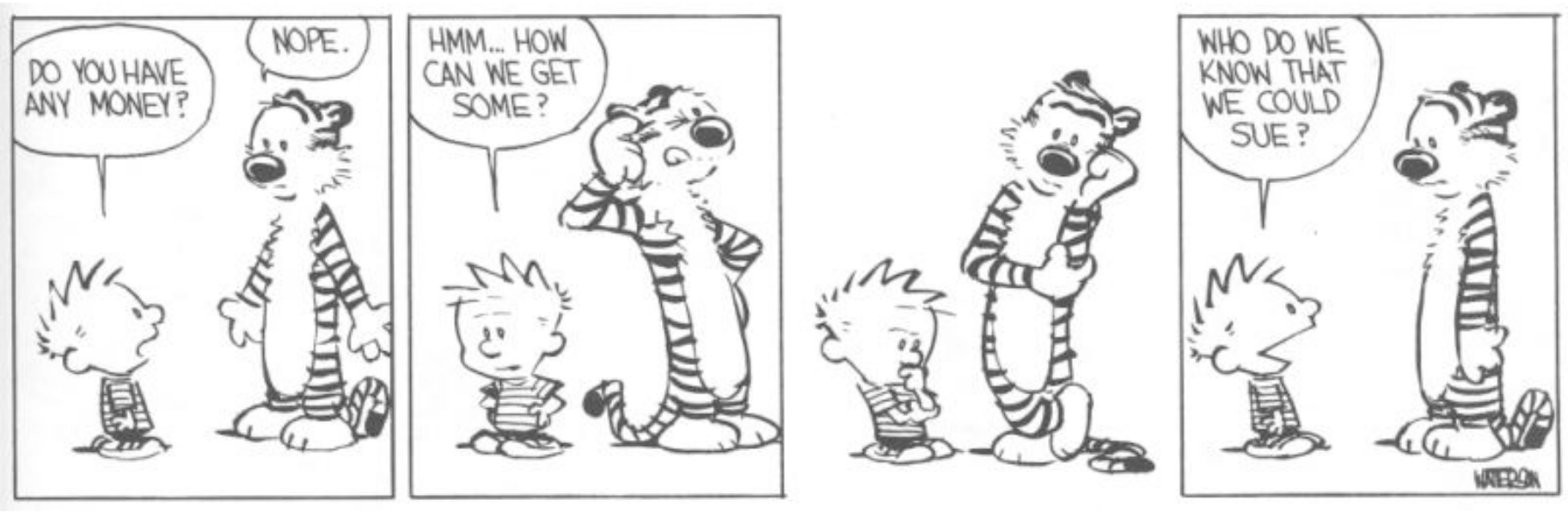
Main Security Specifications

- XML Signature (XMLDSIG)
  - Message Integrity and Sender/Receiver Identification
- XML Encryption (XMLENC)
  - Message Confidentiality
- WS-Security (WSS)
  - Securing SOAP Messages
- SAML
  - Interoperable security metadata exchange
- XACML
  - Access Control

Other Security Specifications

- WS-Trust and WS-Federation
  - Federating multiple security domains
- WS-SecureConversation
  - Securing multiple message exchanges
- WS-SecurityPolicy
  - Describing what security features are supported or needed by a Web service
- XrML
  - Digital Rights Management
- XKMS
  - Key Management and Distribution

XML Signature



XML Signature Overview (1)

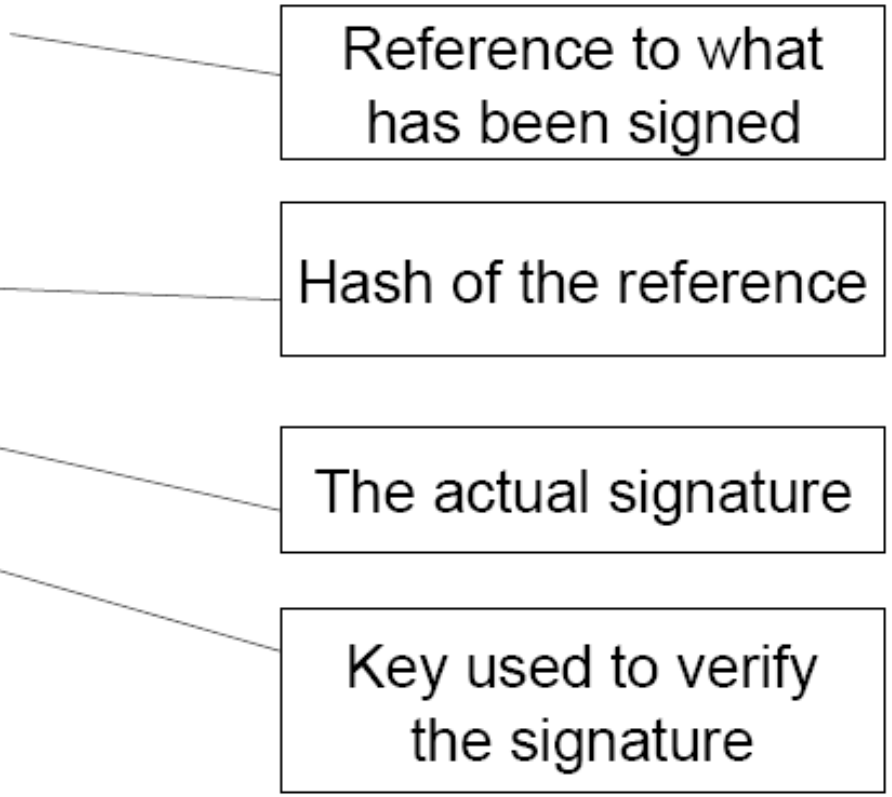
- **Goals**:
  - ensure integrity of XML messages;
  - identify their source/destination
  - ensure non-repudiation
- XML signature prescribes how to compute, store and verify the digital signature of:
  - entire XML documents
  - parts of XML documents
  - “anything that can be referenced from an URL”, this includes non-XML objects, such as Images.

XML Signature Overview (2)

- Complex and flexible standard:
  - It is possible to apply multiple signatures over the same XML content
  - Supports a variety of codes and authentication protocols
- Joint W3C/IETF standard, August 2001

XML Signature Structure

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



XML Signature Simplified Example

```
<Signature>
   <SignedInfo>
     <Reference URI="http://www.google.com"/>
   </SignedInfo>
   <SignatureValue>Base-64 encoded </SignatureValue>
   <KeyInfo>…</KeyInfo>
</Signature>
```

Security Standards Stack (1)

- Reference Generation
  1. Dereference the <Reference URL> to access the XML content that needs to be signed
  2. Apply the Transforms
  3. Compute the <DigestValue> applying the <DigestMethod> to the transformed content
  4. Store the result in the <Reference> element

Security Standards Stack (2)

- Signature Generation
  1. Create the <SignedInfo> element
  2. Transform it to canonical form
  3. Compute the <SignatureValue> applying a <SignatureMethod>
  4. Bundle it all together with the <KeyInfo> and <Object> elements
- Note: what is actually signed is the <Reference>, which contains a digest (hash) of the original content, which is only indirectly signed.

Validating the signature (1)

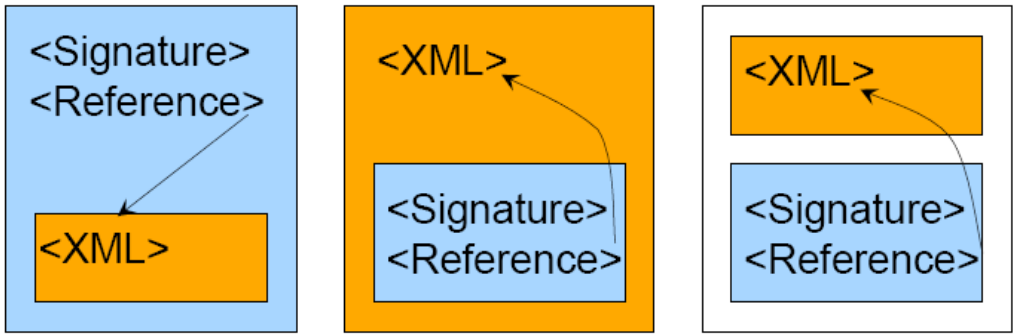
- Reference Validation
  1. Dereference the <Reference URL> to access the XML content that needs to be validated against the digest
  2. Apply the same Transforms
  3. Compute a hash using the same <DigestMethod>
  4. Compare the <DigestValue> with the result.

Validating the signature (2)

- Signature Validation
  1. Canonicalize the <SignedInfo> element
  2. Get the Key following the <KeyInfo> element
  3. Compute the hash with the <SignatureMethod>
  4. Compare it with the <SignatureValue>

XML Signature Position

- Enveloping Signature: the signature wraps the signed element
- Enveloped Signature: the signature is contained inside the signed element
- Detached Signature: the signature refers to a separate element (inside or outside the document)



<Reference> Element

- The reference element points to the resource that is being digitally signed (URI attribute)
- There must be at least one Reference element (but more are possible in the same signature)
- Examples:
  - Hosts An element of the same document URI="#CustomerInformation"
  - The root of the container document URI=""
  - An external XML document URI="http://www.swisscom.ch/order.xml"
  - A fragment of an external document URI="http://www.swisscom.ch/order.xml#Total"
  - An external non-XML resource URI="http://www.swisscom.ch/order.pdf"

<Transformation> Element (1)

- A Reference element contains a set of transform elements, which are applied in a pipelined fashion to the content of the referenced resource
- The same transformations (in the same order) should be used when generating and validating a digest

<Transformation> Element (2)

- Standard Transforms:
  - Canonicalization
  - Enveloped Signature Transform
  - Decrypt Transform
- Optional Transforms:
  - Base-64
  - XPath Filtering
  - XSLT Transform

Canonicalization (C14N)

The problem

- Signatures are sensitive to single bit changes
- XML data can have multiple (and equivalent) serializations. Examples:
  - An XML document from a Windows system will use CR+LF, but can still be parsed in UNIX
  - Whitespace can be represented with TAB
- XML Mismatch between data used by crypto algorithms (raw bytestream: octets) and the XML representation (XML Infoset)

Canonicalization (C14N)

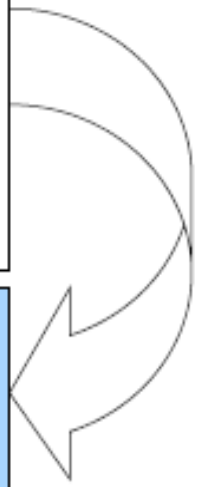
The solution

- Give a precise (and standard) procedure for producing XML “strings” out of XML infosets.
- This procedure is called Canonicalization sensitive to single bit changes

Canonicalization Example

```
<?xml version="1.0"?>
<!DOCTYPE doc SYSTEM "doc.dtd">
<  PurchaseOrder >
   <Customer name = “Swisscom Mobile” />
   <Date            > 2005 11 22 <     /Date>
   <!-- Time unknown -->
   <Items/>
</ PurchaseOrder>
```

Original XML Document



```
<PurchaseOrder>
   <Customer name=“Swisscom Mobile”/>
   <Date> 2005 11 22 </Date>
   <Items></Items>
</PurchaseOrder>
```

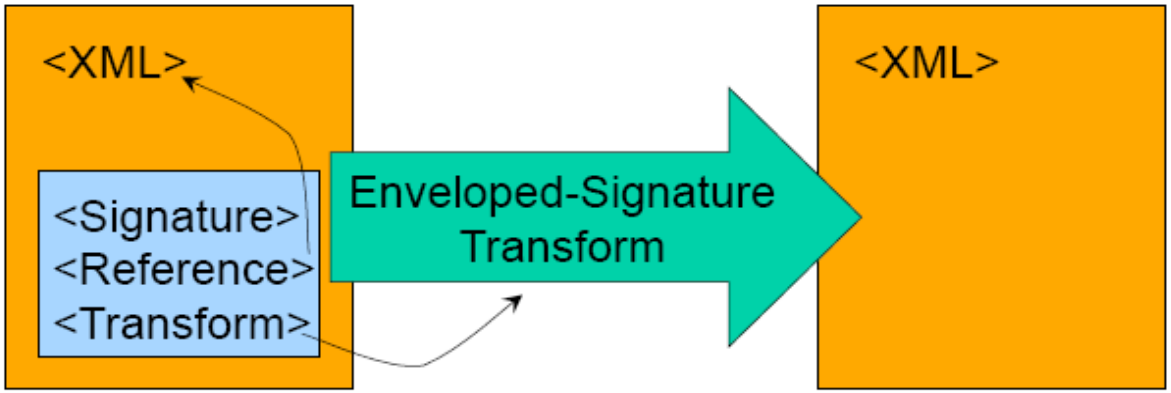
Canonical Form

Some XML Canonicalization Rules

1. BusUTF-8 encoding
2. Linebreaks are normalized to LF (ASCII #xA)
3. Character and entity references are replaced
4. CDATA sections are replaced with their content
5. XML declaration and DTD definition are removed
6. <Empty/> elements converted to <Empty></Empty>
7. Attribute value delimiters are set to double quotes
8. Superfluous namespace declarations are removed
9. Default attributes are explicity added to elements
10. Namespace declarations are sorted before the attributes (also sorted)

Enveloped Signature Transform

- This signature is needed in order to sign an element which is the parent of the <Signature> (Otherwise, the signature should be used as input to compute itself, which makes it impossible to compute)
- This transform simply removes the <Signature> element from the document



Describing and storing the signature

- These elements describe how a signature was computed and store its value in encoded format:
  - The <DigestValue> contains the Base-64 encoded value of the digest
  - The <SignatureValue> contains the Base-64 encoded value resulting from encrypting the digest of the <SignatureInfo> element with the key described in the <KeyInfo>
  - The <DigestMethod> describes the algorithm used to compute the <DigestValue> (e.g., SHA1)
  - The <SignatureMethod> describes how the <SignatureValue> was computed (e.g., RSA-SHA1) using the key

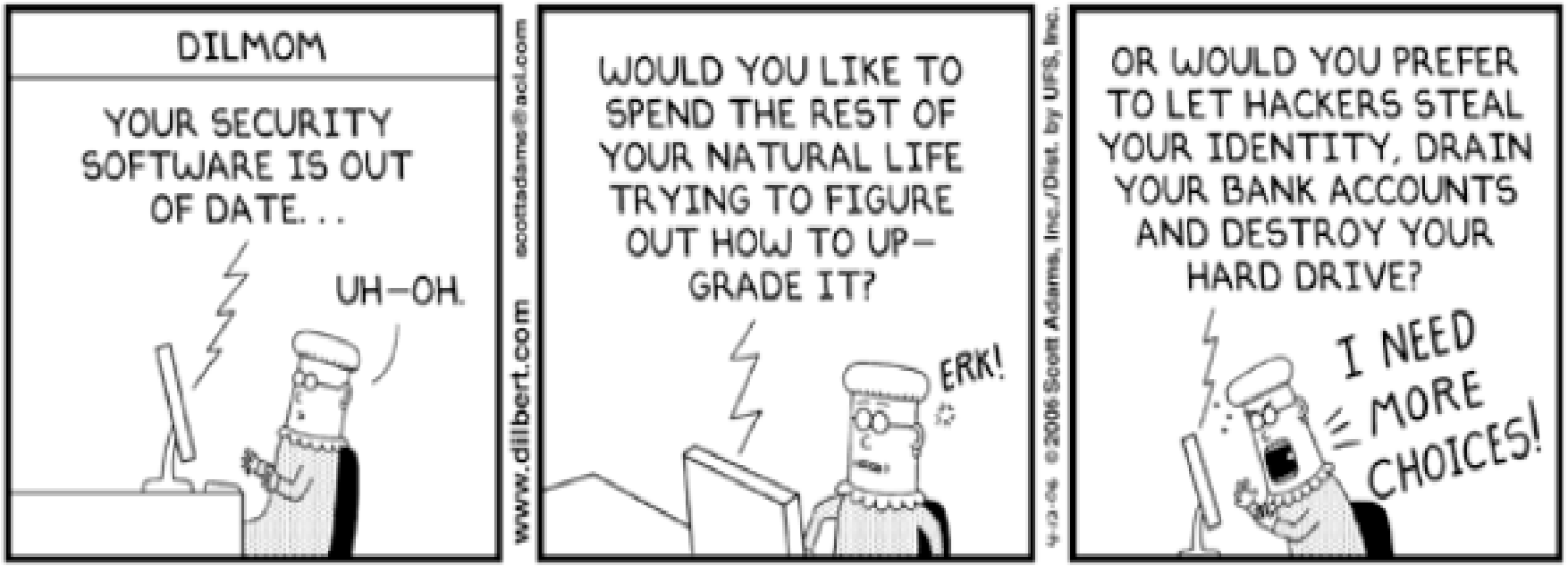
<KeyInfo> element

- The <KeyInfo> provides information about the key used to validate the <SignatureValue>
- It is quite flexible:
  - The element can be omitted (The parties exchanging the message agree on the key using an out-of-band mechanism)
  - Key is embedded in the message
  - Key is referenced from the message
  - It supports several kinds of Keys used with different cryptographic standards:
    - TheDSA/RSA
    - X.509 certificates
    - PGP
- The same element is used in XML Encryption

XML Signature and Security

- XML Signature targets these security aspects:
  1. Integrity of the message content/external resource:
     - Reference validation
  2. Integrity of the signature
     - Signature validation
  3. Identity of the source of the document
     - Signature validation
     - Warning: only if using a <SignatureMethod> based on public/private key
- What you see is what you sign:
  - Transforms modify and filter the data before it is signed, so they should be used carefully

XML Encryption



XML Encryption Overview

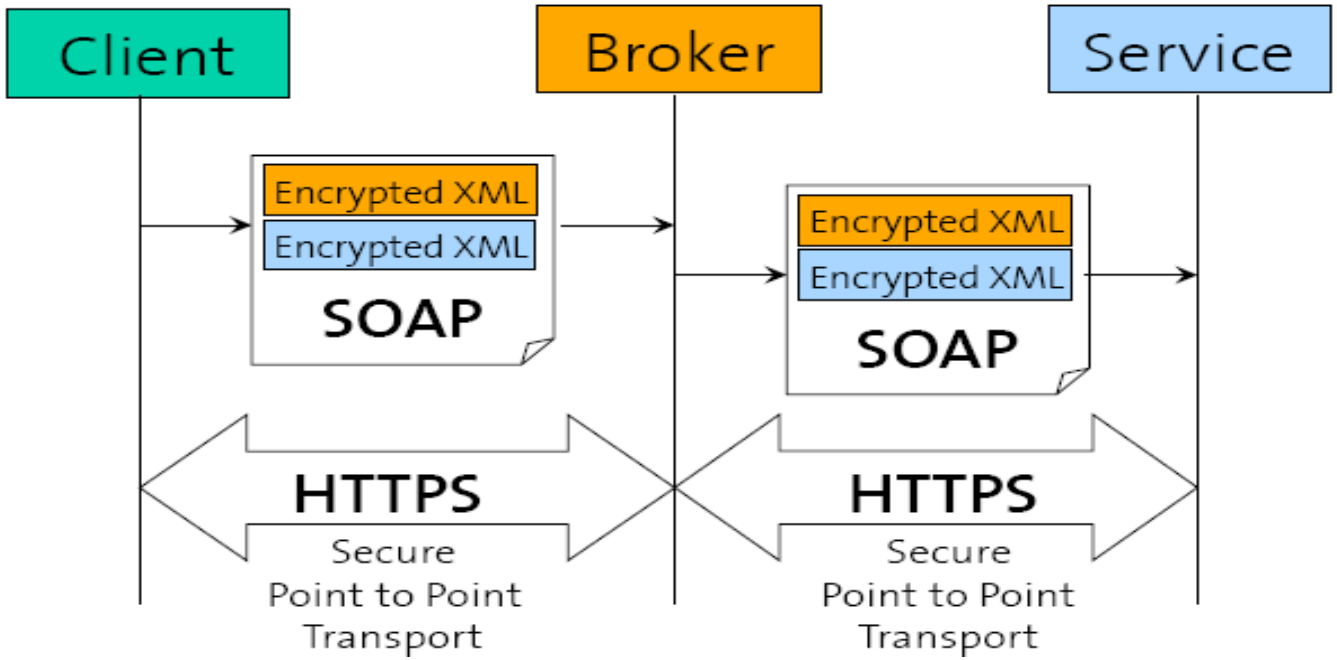
- Goal: ensure confidentiality of XML Messages
- Solution: obfuscate parts of an XML document, while maintaining a correct XML syntax
- Features:
  - End to End (Multi-hop scenario)
  - Full or Partial encryption
  - Flexibility: different parts of a message can be read by different parties using different keys
- Challenges and problems:
  - Is an encrypted XML document still XML?
  - How to validate an encrypted XML document with respect to its XML schema?
- W3C Recommendation, December 2002

XML Encryption vs. **XML Signature**

- XML Encryption complementary to XML Signature
- Different purposes:
  - XML Encryption = Confidentiality
  - XML Signature = Integrity and Identity
- Some overlap in the specifications (e.g., KeyInfo>)
- Difference:
  - XML Encryption. Encrypted XML is replaced by the <EncryptedData> element
  - XML Signature: Signed XML is referenced from the <Signature> element
- Warning: Encrypted data which is not signed can still be tampered with!

XML Encryption Scenario

• Guarantee confidentiality at the SOAP message level (Selected parties may access different message parts)



XML Encryption Example

Original XML Document

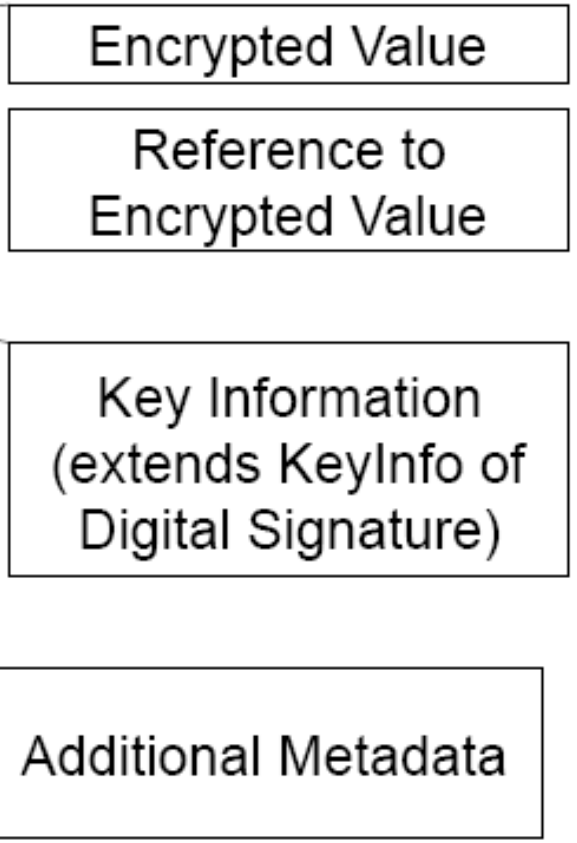
```
<Employee>
    <ID>222-654-456</ID>
    <Name>Markus Bach</Name>
    <Salary currency="CHF">100000</Salary>
</Employee>
```

Encrypted XML Document

```
<Employee>
    <ID><EncryptedData>...</EncryptedData></ID>
    <Name>Markus Bach</Name>
    <EncryptedData>...</EncryptedData>
</Employee>
```

XML Encryption Structure

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



<EncryptedData> Element (1)

- The <EncryptedData> container tag replaces the document elements that are sent in encrypted form
- Together with the encrypted elements <CipherData>, it contains metadata and attributes describing how to decrypt them <EncryptionMethod>,<KeyInfo>
- Attributes:
  - Type = (element | content). Determine whether the plaintext is an entire XML element or only the content has been encrypted.
  - MimeType. Optional attribute describing the type of the encrypted non-XML element
  - Encoding. How the non-XML has been encoded

<EncryptedData> Element (2)

- The <EncryptionMethod> specifies which algorithm has been used to encrypt the data. Currently supported are:
  - Triple-DES
  - AES (Advanced Encryption Standard) with 128, 256 (required) or 192 (optional) bit key