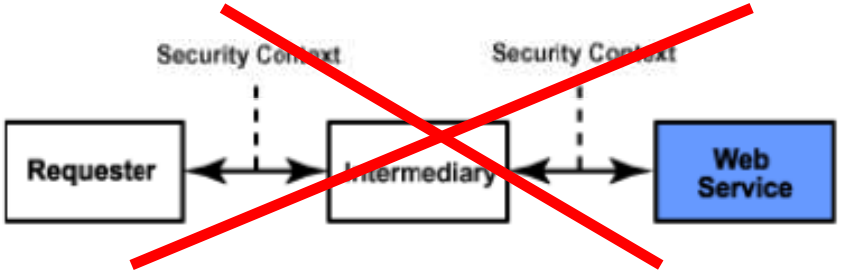
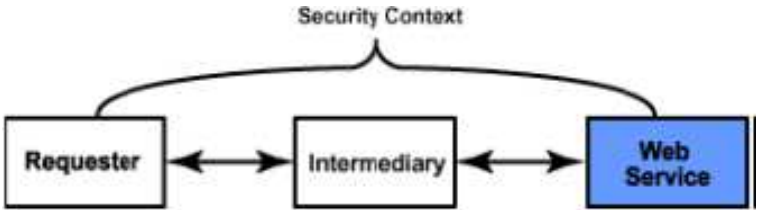
# WS-Security

# Motivation

- Transport Layer Security (TLS) can provide point-to-point security, but not end-to-end, problem with proxies




- Want:
  - Message integrity
  - Message confidentiality



# The WS-Security solution

- ~ two years after Web Services was introduced, IBM, Microsoft and VeriSign addressed the security issue.
- In April 2002 they released the proposed specification for WS-Security from SOAP-Security, WS-Security, WS-License
- April 2004: The standard was released as WS-Security 1.0 by Oasis-Open
- February 2006: Oasis-Open released “Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)” or WS-Security Core Specification 1.1

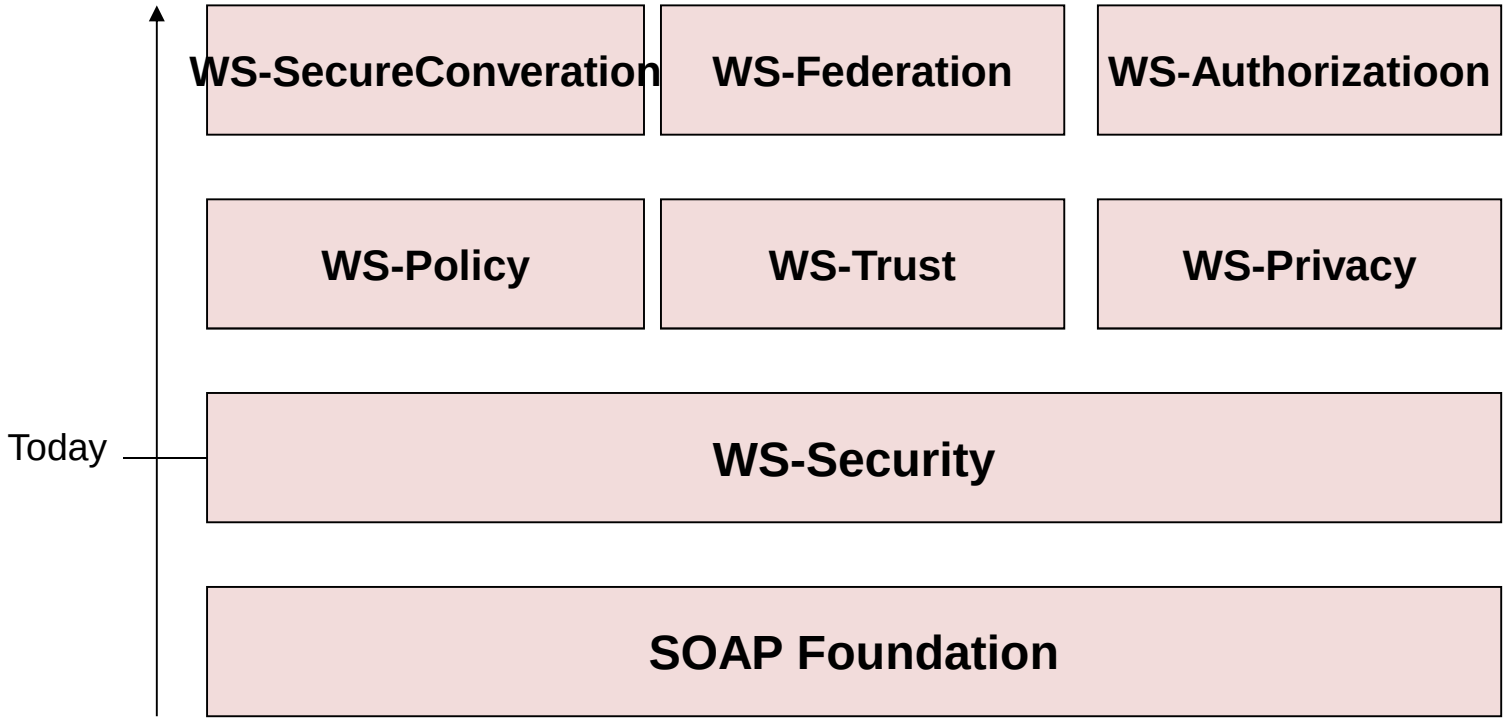
# WS-Security 1.1

From the spec:

- Enhancements to SOAP to provide integrity and confidentiality
- Accommodates a wide variety of security models and encryption technologies
- Provides a mechanism for associating security tokens with message content
- Of course extensible: Supports multiple security token formats, can define different formats for different parts of the message

# Web Services Security Specifications

- The combination of security specifications, related activities, and interoperability profiles will enable customers to easily build interoperable secure Web services.



# Disclaimer

- Provides flexible set of mechanisms to **construct** a range of security protocols
- Does not describe explicit fixed security protocols
- This means: It is up to you to design your non-vulnerable protocol

# Goals of the specification

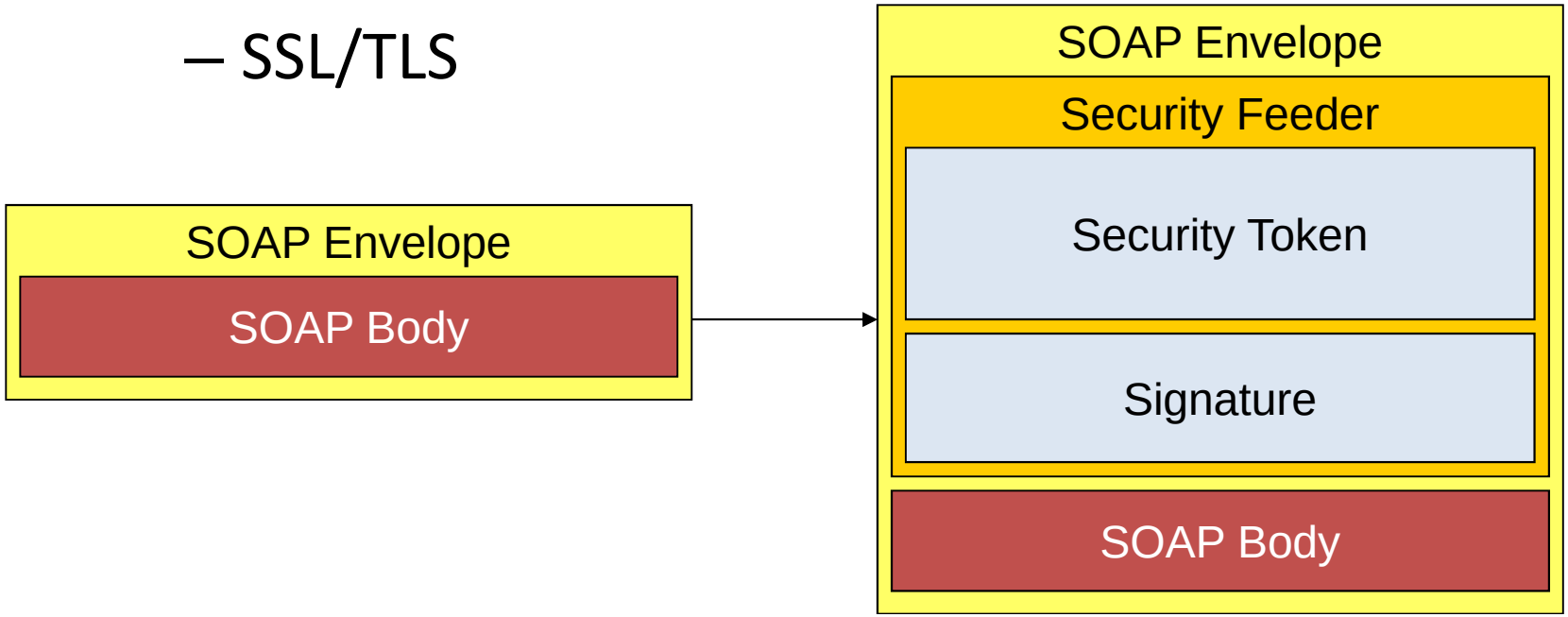
- Multiple security token formats
- Multiple trust domains
- Multiple signature formats
- Multiple encryption technologies
- End-to-end message **content** security

# (Terminology)

- **Confidentiality** – the property that data is not made available to unauthorized individuals, entities, or processes (encryption)
- **Integrity** – the property that data has not been modified (signature)
- **Claim** – a declaration made by an entity (e.g. name, identity, key, group, privilege, capability, etc).
- **Claim Confirmation** – the process of verifying that a claim applies to an entity.
- **Security Token** – represents a collection of claims.
- **Signed Security Token** – a security token that is asserted and cryptographically signed by a specific authority (e.g. an X.509 certificate or a Kerberos ticket).
- **Trust -** the characteristic that one entity is willing to rely upon a second entity to execute a set of actions and/or to make set of assertions about a set of subjects and/or scopes.

# WS-Security

- Enhancement to SOAP
- Uses
  - XML Encryption xmlenc
  - XML Digital Signatures xmlsig
  - SSL/TLS



# Security Header Block

- No blocks with same S11:actor or S12:role
- Only one may omit actor/role attribute

For extensibility, should be based on schema:

/wsse:Security/{any}

/wsse:Security/@{any}

```
<S11:Envelope>
<S11:Header>
...
<wsse:Security S11:actor="..." S11:mustUnderstand="...">
...
</wsse:Security>
<wsse:Security S12:role="..." S12:mustUnderstand="...">
…
</wsse:Security>
...
</S11:Header>
...
</S11:Envelope>
```

(see WSS 1.1 Spec pg 21 for description)

# SOAP Example

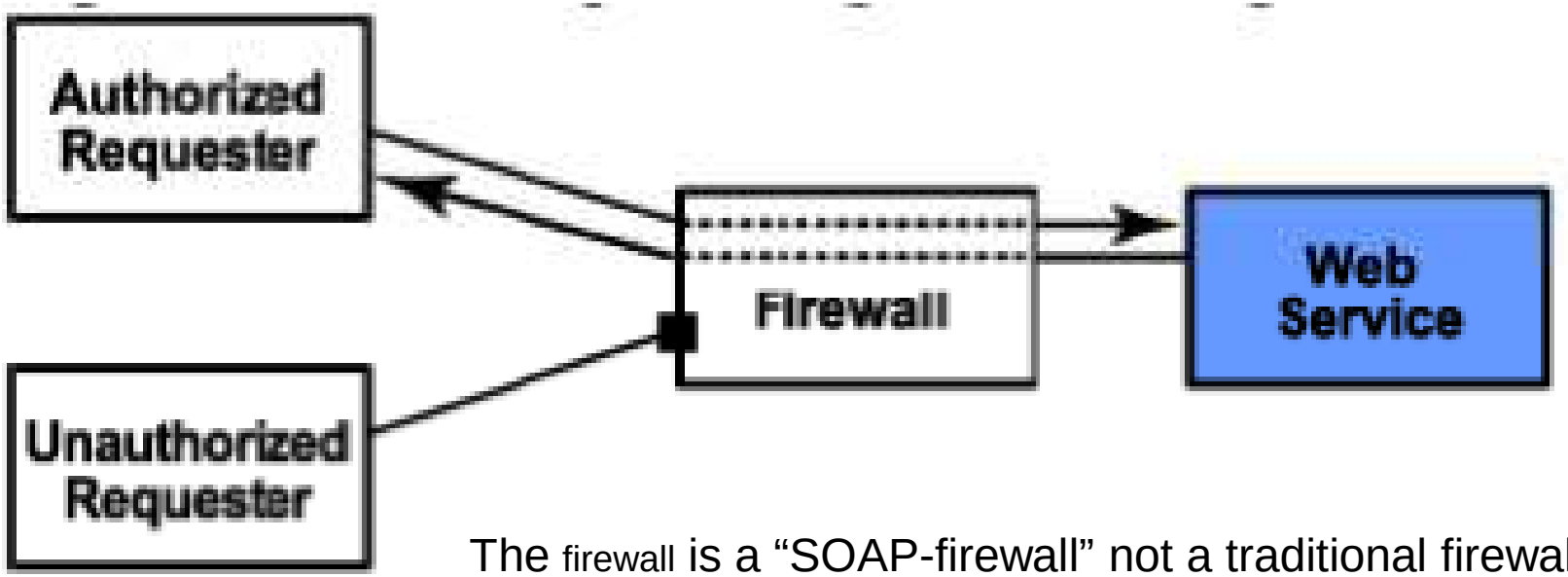
```
<?xml version="1.0" encoding="utf-8"?>
<S11:Envelope  xmlns:S11="...">
    <S11:Header>
    </S11:Header>
    <S11:Body wsu:Id="MsgBody">
          <tru:StockSymbol
    xmlns:tru="http://fabrikam123.com/payloads">
                    RCOM
          </tru:StockSymbol>
    </S11:Body>
</S11:Envelope>
```

# WSS'ed SOAP Example

```
<?xml version="1.0" encoding="utf-8"?>
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="...“ xmlns:ds="...">
 <S11:Header>
  <wsse:Security xmlns:wsse="...">
   <wsse:BinarySecurityToken ValueType=" http://fabrikam123#CustomToken " EncodingType="...#Base64Binary" wsu:Id=" MyID
     ">
    FHUIORv...
   </wsse:BinarySecurityToken>
   <ds:Signature>
    <ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
     <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
     <ds:Reference URI="#MsgBody">
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>LyLsF0Pi4wPU...</ds:DigestValue>
     </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>DJbchm5gK...</ds:SignatureValue>
    <ds:KeyInfo>
     <wsse:SecurityTokenReference>
      <wsse:Reference URI="#MyID"/>
     </wsse:SecurityTokenReference>
    </ds:KeyInfo>
   </ds:Signature>
  </wsse:Security>
 </S11:Header>
 <S11:Body wsu:Id="MsgBody">
  <tru:StockSymbol xmlns:tru="http://fabrikam123.com/payloads">
   RCOM
  </tru:StockSymbol>
 </S11:Body>
</S11:Envelope>
```

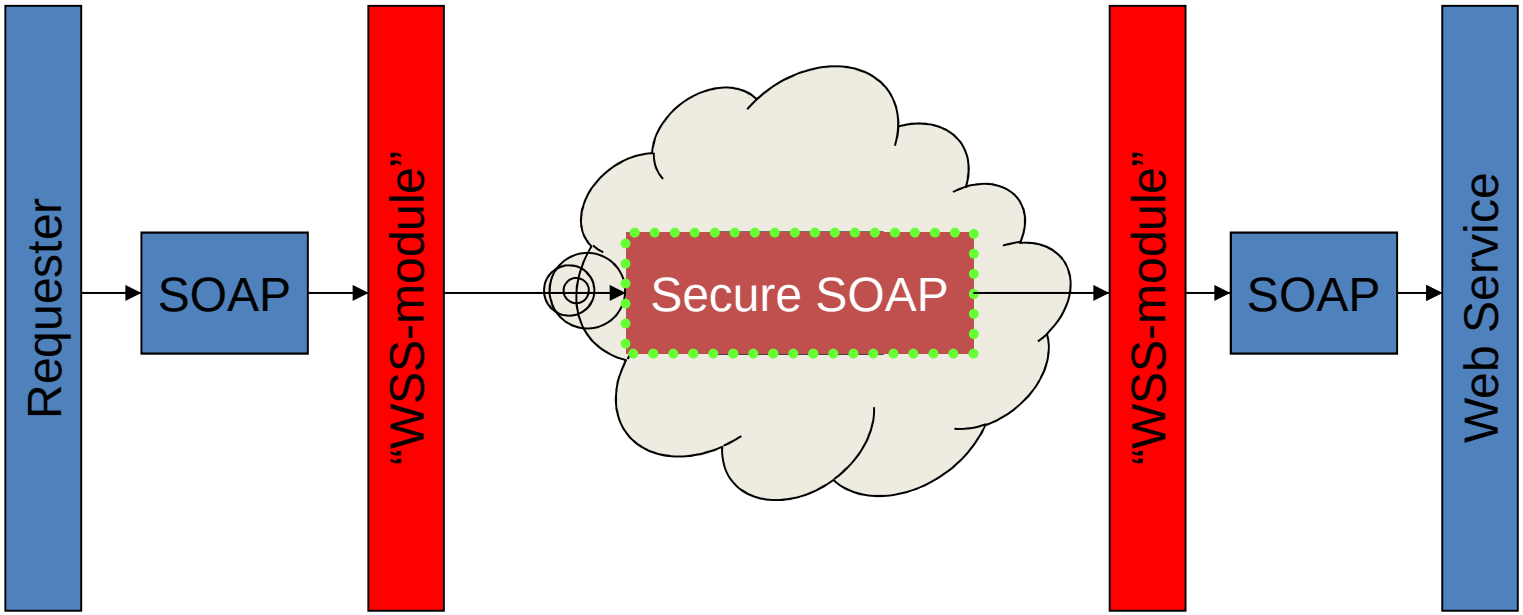
(see WSS 1.1 Spec pg 15 for description)

# Enables the scenario




The firewall is a “SOAP-firewall” not a traditional firewall

# …and this:


Requester
SOAP
“WSS-module”
Secure SOAP
“WSS-module”
SOAP
Web Service

# You don’t need to code XML

```
public Message signSOAPEnvelope(SOAPEnvelope unsignedEnvelope)
    throws Exception
  {
    // WSSignEnvelope signs a SOAP envelope according to theWS Specification (X509 profile) and adds the signature
    // data to the envelope.
    WSSignEnvelope signer = new WSSignEnvelope();

    signer.setUserInfo("16c73ab6-b892-458f-abf5-2f875f74882e", "foobar");

    Document doc = unsignedEnvelope.getAsDocument();

    // The "build" method, creates the signed SOAP envelope. It takes a SOAP Envelope as a W3C Document and
    // adds a WSS Signature header to it. The signed elements depend on the signature parts that are specified by
    // the WSBaseMessage.setParts(java.util.Vector parts) method. By default, SOAP Body is signed.
    // The "crypto" parameter is the object that implements access to the keystore and handling of certificates.
    // A default implementation is included: org.apache.ws.security.components.crypto.Merlin
    Document signedDoc = signer.build(doc, crypto);

    // Convert the signed document into a SOAP message.
    Message signedSOAPMsg = (org.apache.axis.Message)AxisUtil.toSOAPMessage(signedDoc);

    return signedSOAPMsg;
  }
```

# Transforms this:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-
    envelope" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
    <sayHello xmlns="http://jeffhanson.com/services/helloworld">
            <value xmlns="">
                    Hello world!
            </value>
    </sayHello>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# Into this:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header>
  <wsse:Security SOAP-ENV:mustUnderstand="true" xmlns:wsse="http://docs.oasis-open.org/...-wss-wssecurity-secext-1.0.xsd">
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:SignedInfo>
           <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
            <ds:Reference URI="#id-1281123">
              <ds:Transforms>
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                </ds:Transforms>
                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <ds:DigestValue>wLumPkKZ+X48rjao/XUUQDp0xk0=</ds:DigestValue>
            </ds:Reference>
          </ds:SignedInfo>
        <ds:SignatureValue>a56OxPcKr8LJnIFgRyMQej5/ZkUjkV9V9rmn+queMKzJ3GYpMiXpjQ==</ds:SignatureValue>
        <ds:KeyInfo Id="KeyId-30752603">
          <wsse:SecurityTokenReference wsu:Id="STRId-2545159" xmlns:wsu="http://docs...-200401-wss-wssecurity-utility-1.0.xsd">
            <ds:X509IssuerSerial>
              <ds:X509IssuerName>CN=pubcert</ds:X509IssuerName>
              <ds:X509SerialNumber>1140726843</ds:X509SerialNumber>
            </ds:X509IssuerSerial>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
    </ds:Signature>
  </wsse:Security>
</SOAP-ENV:Header>
<SOAP-ENV:Body wsu:Id="id-1281123" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <sayHello xmlns="http://jeffhanson.com/services/helloworld">
        <value xmlns="">Hello world!</value>
        </sayHello>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# ID References

- WSS defines the wsu:Id attribute, type xsd:ID

`<anyElement wsu:Id="…">…</anyElement>`

- Used to locate elements in the message e.g. correlating signatures to sec. tokens
- XML Schema defines several id and referencing data types, but they require consumer to have or obtain schema definition.
- For intermediaries this can be "heavy" and not desirable
- May also use `<wsse:SecurityTokenReference>` for referencing security tokens

# Security Tokens – User Name

- Introduced as a way to provide username
- Optional

For extensibility,
should be based
on schema:
/wsse:UsernameToken/@wsu:Id
/wsse:UsernameToken/
wsse:Username
/wsse:UsernameToken/{any}
/wsse:UsernameToken/@{any}

```
<S11:Envelope xmlns:S11="..."
xmlns:wsse="...">
  <S11:Header>
    ...
    <wsse:Security>
      <wsse:UsernameToken>

<wsse:Username>Zoe</wsse:Username>
      </wsse:UsernameToken>
    </wsse:Security>
    ...
  </S11:Header>
  ...
</S11:Envelope>
```

(see WSS 1.1 Spec pg 21 for description)

# Security Tokens – BinarySecurityToken

- Binary formatted security tokens, X.509, Kerberos ticket or other non-XML formats

@ValueType – e.g. Kerberos or X.509

@EncodingType – e.g. Base64Binary (deflt)

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

/wsse:BinarySecurityToken/@{any} for additional attributes

(see WSS 1.1 Spec pg 25 for description)

# SecurityTokens - EncryptedData

<xenc:EncryptedData>

This element may be used to contain a security token and included in <wsse:Security> header

When processed, it is replaced with it's decrypted form in message

```
<wsse:Security>
   <xenc:EncryptedData …>…</xenc:EncryptedData>…
```

Becomes:

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

Can also be used to encrypt other elements

# WS-Security Drawbacks

- WS-Security defines mechanisms for enabling integrity and confidentiality for Web Service messages.
- However, if the corresponding WS-SecurityPolicy is not defined correctly, attacks on integrity and confidentiality are possible using so-called XML rewriting attacks
- WS-Security does not define any direct countermeasures against attacks like Denial-of-Service.

# References


- http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
- http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf
- http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf
- http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf
- http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf
- http://www.oasis-open.org/committees/download.php/16687/oasis-wss-rel-token-profile-1.1.pdf
- http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-spec-os-SwAProfile.pdf
- http://www-128.ibm.com/developerworks/library/specification/ws-secmap/
- http://www-106.ibm.com/developerworks/webservices/library/ws-secure/
- http://www.pentrix.com/videos/videolist.php
- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-security-appnote.asp
- http://www.codeproject.com/webservices/WS-Security.asp
- http://www.devx.com/Java/Article/28816/1954?pf=true