IBM Software Group

Understanding Web Services Security

Mark Colan, Evangelist, SOA and Web Services
Jeff Miller, Senior e-business Architect

Download PDFs of presentations on SOA and Web services
http://ibm.com/developerworks/speakers/colan

e business on demand software

Agenda

- Concepts: Security goals for e-business applications
- General security technologies
- Web services security
- Emerging Web services security technologies
- Security in today's Web services products
- Resources

Security Goals and Requirements

There is no such thing as absolute security

- There are risks and countermeasures to address these risks

Nothing can ever be proven to be 100% secure

- But we can make the cost of breaking into a system more expensive than the value of the information it contains

Security requirements vary with different applications

- There's no universal checklist
- There are common requirements, but they may not all apply

Security must be based on strong, **open standards** to ensure interoperability between platforms

- We need a standard that says how these security standards can be used for Web services

Seven ISO Security Requirements

Identification: who are you?

Authentication: how do I know your identity is true?

Authorization: are you allowed to perform this transaction?

Integrity: is the data you sent the same as the data I received?

Confidentiality: are we sure no one read the data you sent me?

Auditing: record of all transactions so we can look for security problems after the fact

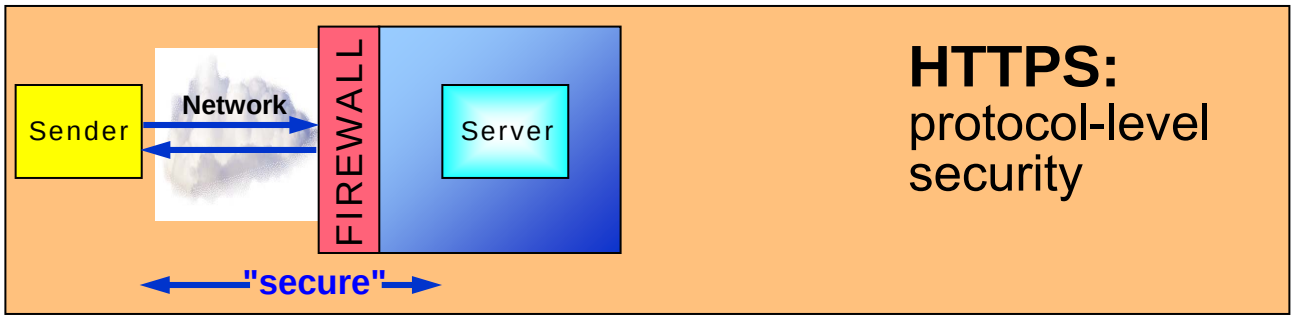
Non-repudiation: both sender and receiver can provide legal proof to a third party (e.g. judge) that

- the sender did send the transaction, and
- the receiver received the identical transaction

Using HTTPS/SSL for SOAP Message Security

HTTPS/SSL provides "protocol-level" or "transport-level" security

- **Identification**, basic **authentication**, **encryption**, and built-in **integrity** check
- Point-to-point security across one connection
- Convenient: application merely asks for secure connection
- No other application requirements



Limitations of HTTPS in SOAP messaging

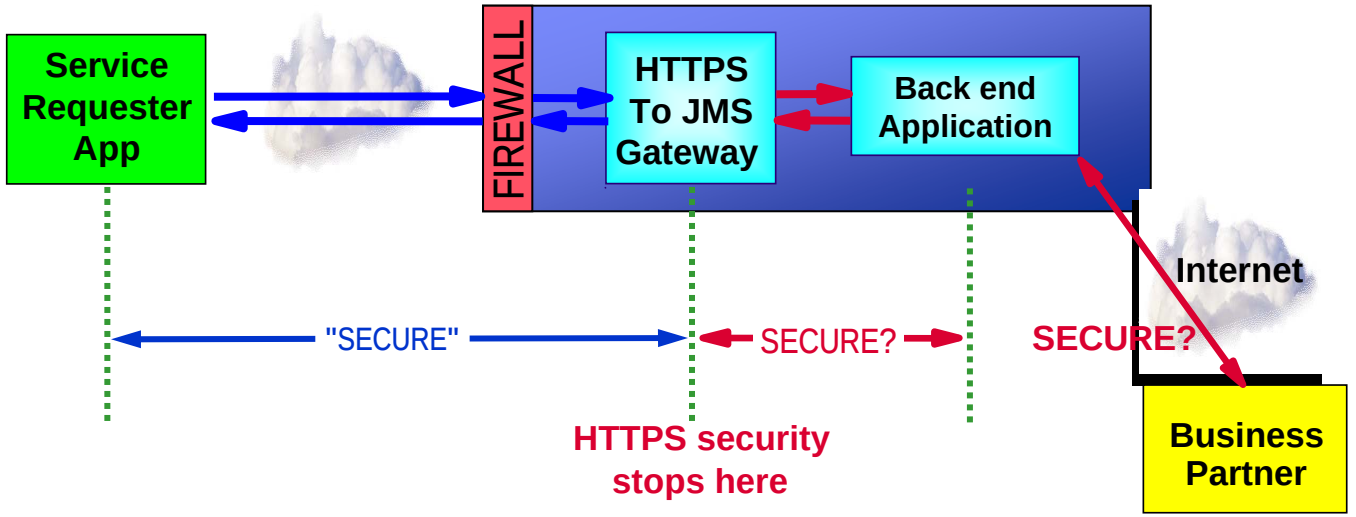
Limitation 1: HTTPS offers **Point-to-point security** only

- Security stops at HTTPS endpoint
- We need **end-to-end security** across all connections in the life of a transaction for an effective business environment.

Limitation 2: No Digital Signature

- No means of conveying other kinds of security tokens

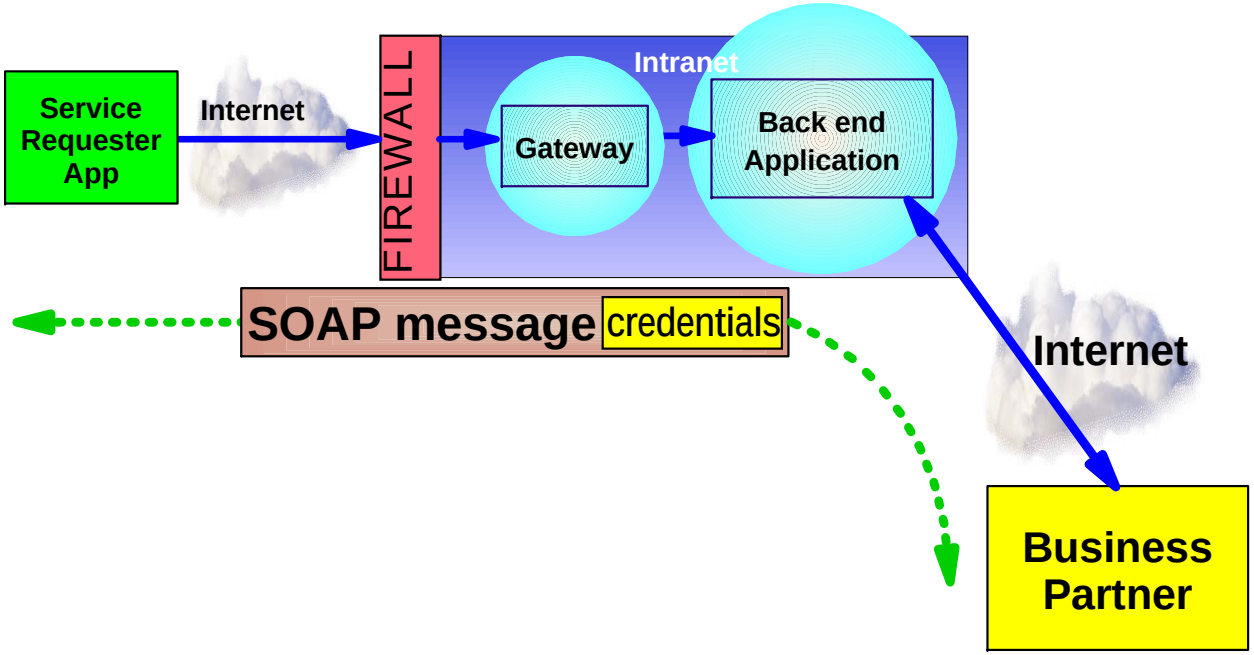
Limitation 3: HTTPS all-or-nothing encryption



Message-level Security for SOAP messages

By placing security information **in the message itself**, we can overcome these limitations and achieve an end-to-end solution

To ensure interoperability, we need a standard to define new security elements to extend the SOAP message and allow the use of both proven and emerging security technologies



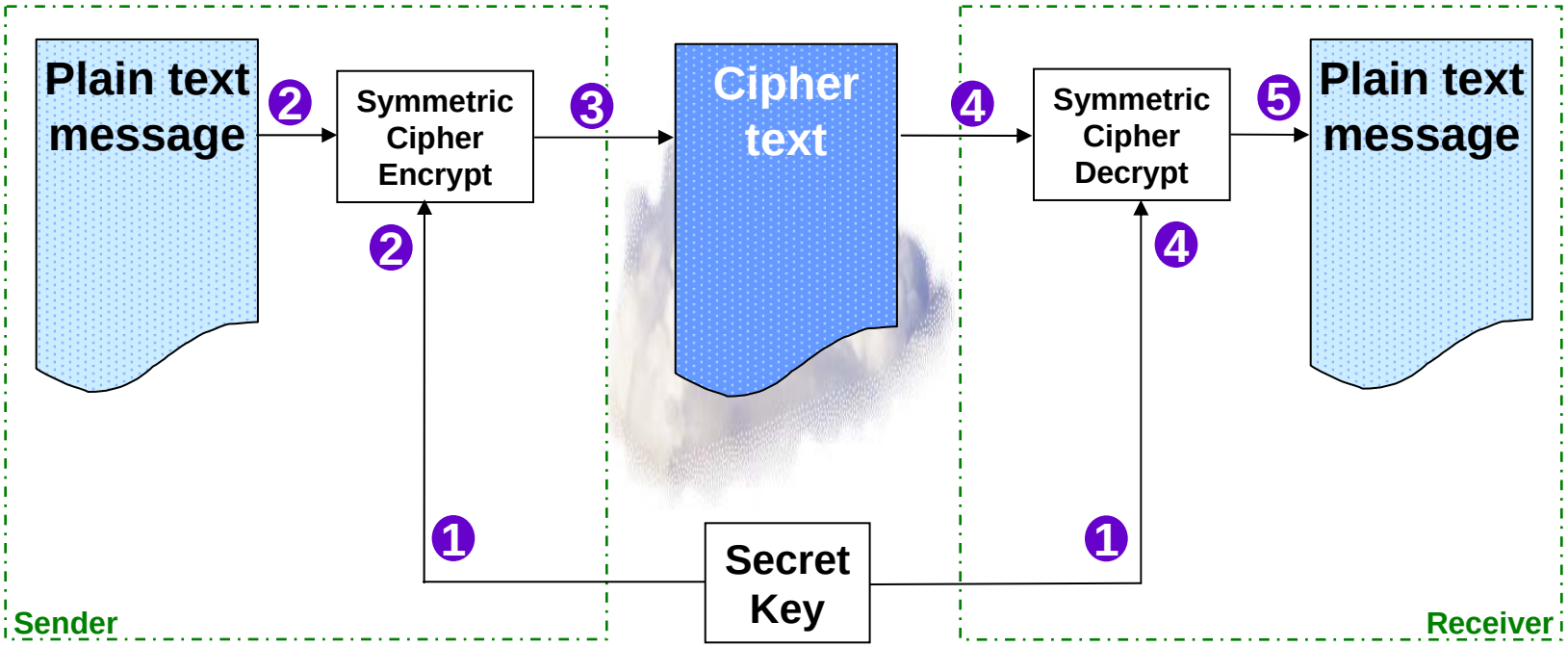
Core Security Technologies

Let's discuss some basic technologies for security:

- Symmetric encryption
- Asymmetric encryption
- XML Encryption
- Hash functions
- XML Digital Signature
- Digital certificates

Symmetric Encryption

The same secret key is used to both encrypt and decrypt the message



Symmetric Encryption

Fast

Common algorithms are Triple DES (3DES), AES, …

Drawback: the key must remain secret, and it must be distributed securely to anyone we want to talk with

- If we want secure conversations with n partners, we have to distribute n keys to them

If the partner is local, we can hand them the key on any convenient digital media

But if they are distant, this isn't convenient, and we can't safely send it to them using the Internet!

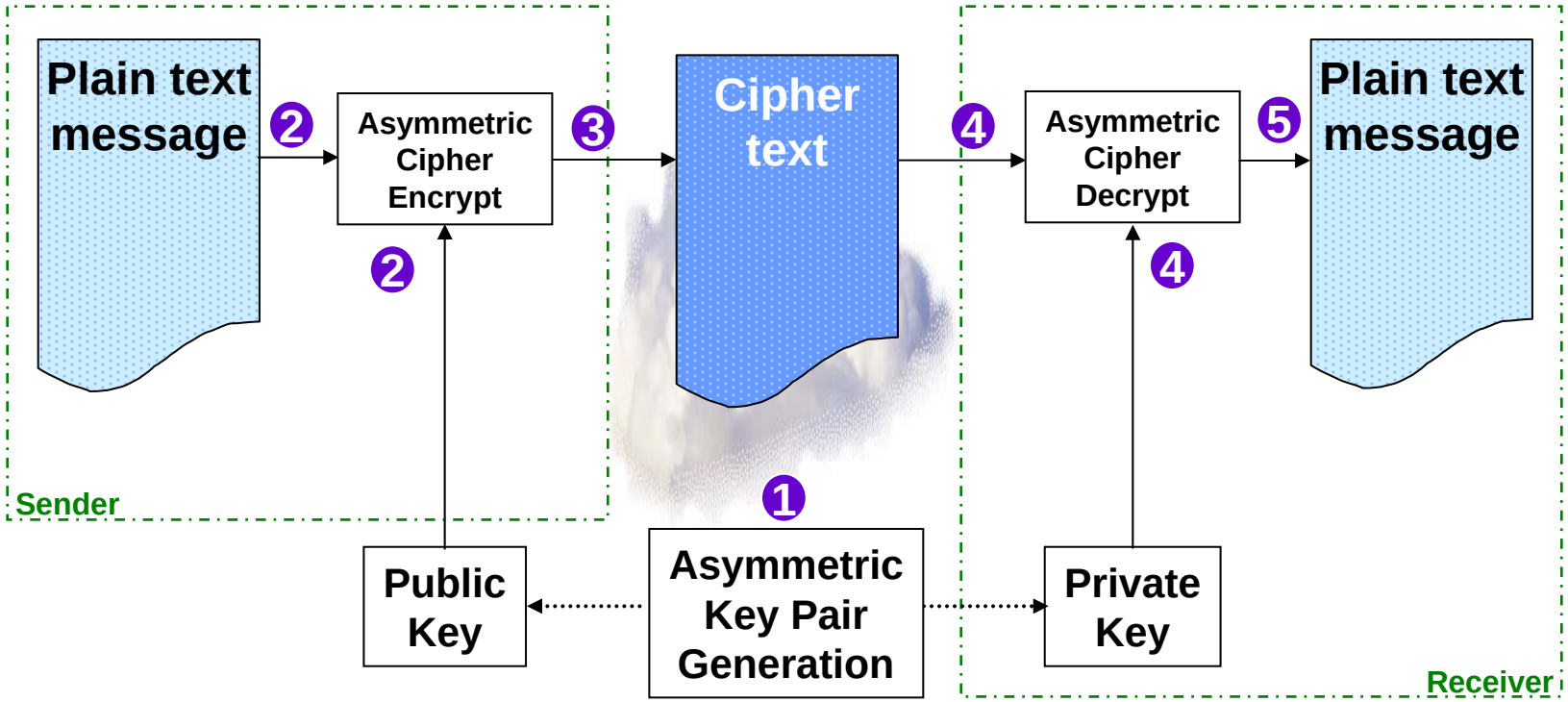
Asymmetric Encryption

Each owner has a pair of complementary, asymmetric keys

- They are different from each other
- Encrypt with one, decrypt only with the other (in either direction)
- We give one away (the Public key) and keep the other secret (the Private key)
- If anyone encrypts a message with our public key, only we can decrypt the message (with our private key)
- Conversely, if we encrypt a message with our private key, only our public key will decrypt it. So…
  - If a recipient successfully decrypts that message with our public key, they know we sent the message
- Drawback: asymmetric encryption is slower than symmetric encryption

Asymmetric Encryption

Encrypt with the receiver's Public Key -- only the receiver can decrypt the message



1 Public and private keys belong to the receiver

Public Key Cryptography, the basis for PKI

Hybrid approaches

Cryptography applications frequently combine these approaches

- SSL is an example: It uses asymmetric encryption to setup the secret key, then uses symmetric encryption for the data transmission

Most of the ISO security issues are solved with protocols based on combinations of symmetric encryption, asymmetric encryption, and hash functions

XML Encryption

The XML Encryption standard defines ways to encrypt all or part of an XML document

- The encrypted information is replaced with a single **<EncryptedData>** element
- You can encrypt different parts of the same document with different keys
- You can encrypt the whole document, a single element, or just the text of an element

What's in <EncryptedData>

An **<EncryptedData>** element contains these elements

- **<EncryptionMethod>** – The algorithm used to encrypt the data
- **<KeyInfo>** – Information about the key used to encrypt the data
- **<CipherData>** – Contains the
  - **<CipherValue>** element, which in turn
    - Contains the actual encrypted data

As we'll see shortly, XML encryption in the context of Web services changes the format a little

W3C XML Encryption specifications

Who: W3C Working Group

- http://www.w3.org/Encryption/
- Started as joint proposal by IBM, Microsoft, Entrust

Purpose:

- Encrypting data and representing the result in XML
- Can encrypt: an entire XML document, elements, element content, arbitrary data, or a combination of these
- **<EncryptedData>** replaces encrypted element or content, or is the root of an encrypted document

Status: W3C Recommendations, December 2002

- XML Encryption Syntax and Processing 1.0
- Decryption Transform for XML Signature 1.0

Availability:

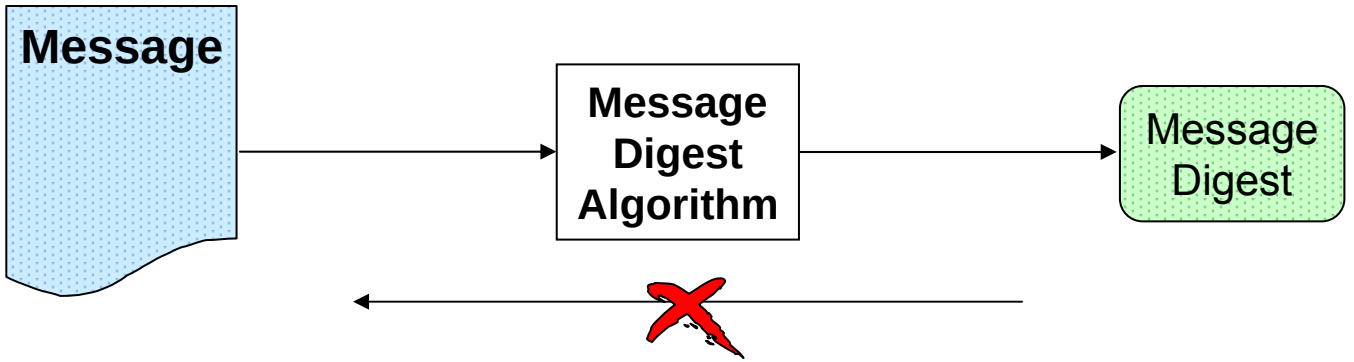
- WebSphere Application Server v4, v5, free IBM WSDK
- Apache XML Security project: http://xml.apache.org/security/index.html

Hash functions

A hash or message digest function reduces an arbitrary stream of bytes to a fixed-size number (usually 128 or 160 bits)

It has two important properties:

1. **Given an input stream and its hash code, it's practically impossible to find a second stream with the same hash code**
2. **A small change to the original input stream produces a huge change in the hash code**



XML Digital Signature

Provides proof of **integrity** of XML content

- The signed data has not changed since it was sent
- Does NOT provide confidentiality

Based on hash functions and encryption

1. Generate a hash from the data to be signed
2. Encrypt the digest to create the signature
3. The signature is sent with original content for verification purposes

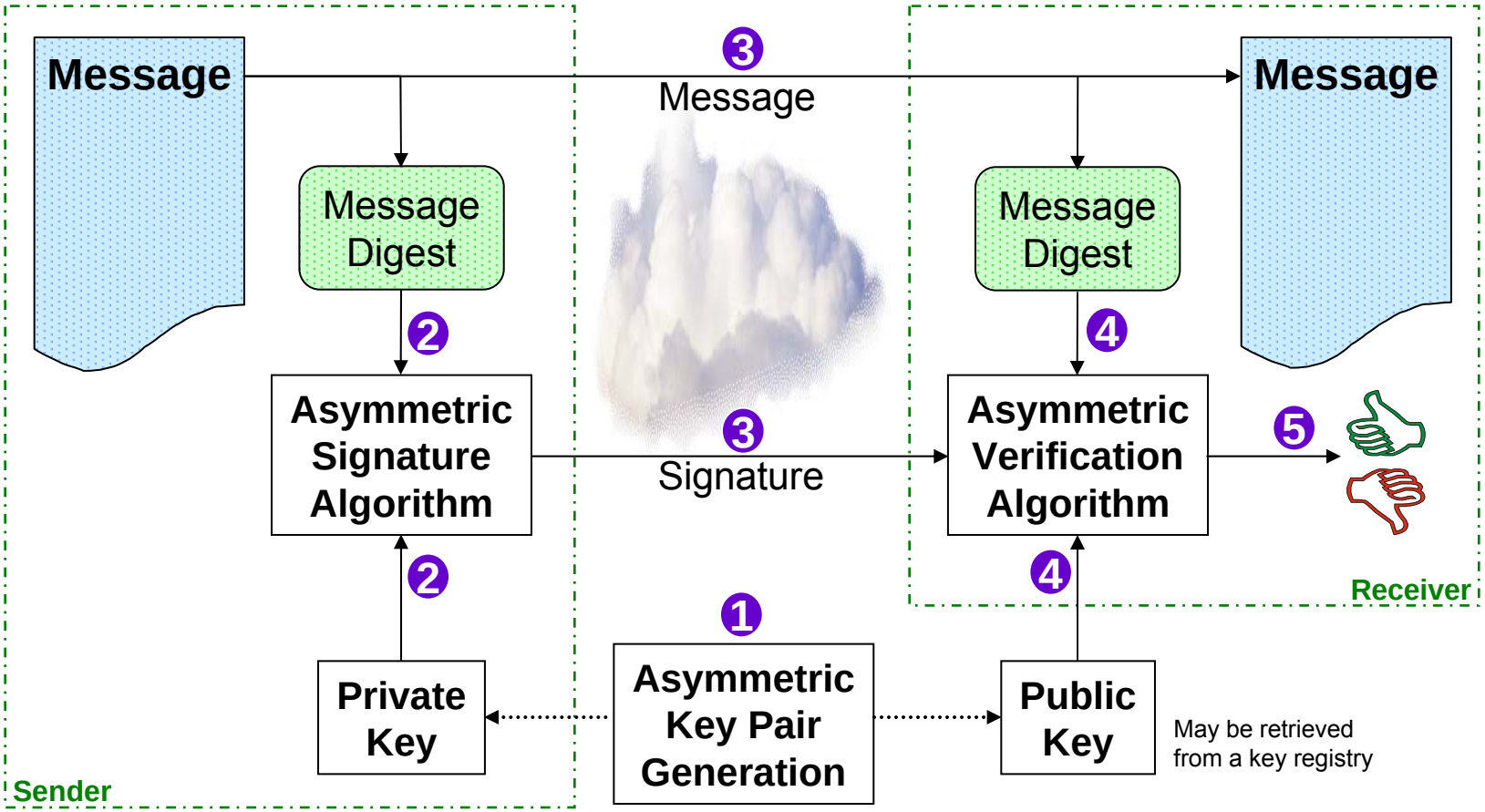
To verify the signature

1. Regenerate a digest of the original data that was signed
2. Decrypt the first encrypted digest (i.e. the signature)
3. Compare the two digests; a match verifies the content

Along with Auditing, XML Digital Signature gives us Non-repudiation

We'll look at signatures from a Web services perspective in a moment

General Digital Signature Processing



1. Public and private keys belong to the sender
3. Signature appended to message and sent
4. Compute new digest, decrypt signature, compare, valid if equal

XML Digital Signature

An XML digital signature is stored in a **<Signature>** element

It has three main parts

- **<SignedInfo>** – Information about what is signed
- **<SignatureValue>** – The value of the digital signature itself
- **<KeyInfo>** – The public key used to verify the signature

Steps

- Calculate a **<DigestValue>** and create **<Reference>** elements for data to be signed
- Add **<DigestValue>** and **<Reference>** elements into **<SignedInfo>**
- Sign the entire **<SignedInfo>** element to create a **<SignatureValue>** element
- Add **<SignedInfo>**, **<SignatureValue>**, and **<KeyInfo>** to **<Signature>**

Example: XML Signature (no SOAP)

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

  <SignedInfo>                                              What is signed?
    <CanonicalizationMethod   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <SignatureMethod    Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <Reference    URI="#wssecurity_body_id_26012129343116680961040651106378">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>AWQKpmksMpzzT4PxcizO980gVHw=</DigestValue>
    </Reference>
  </SignedInfo>

                                                            Signature Value
  <SignatureValue>bNhT+DsNN9PR [binary data has been truncated]</SignatureValue>

  <KeyInfo>                                                 Public Key (optional)
    <wsse:SecurityTokenReference>
      <wsse:Reference URI="#wssecurity_binary_security_token_id_1603091_4272645"/>
    </wsse:SecurityTokenReference>
  </KeyInfo>

</Signature>                                                Signature Block
```

XML Digital Signature

The XML Digital Signature standard defines rules for creating a digital signature and storing that signature inside an XML document

XML-Signature Syntax and Processing 1.0

- http://www.w3.org/Signature/
- Definition of schema for the signature (KeyInfo)
- Procedures for computing and verifying such signatures
- Signature survives parsing/generation operations
- Sign entire document, portions, or combinations of these

Status: W3C Recommendation, February 2002

Related specification: XML Exclusive Canonicalization

- Specifies order of processing in computing a signature
- http://www.w3.org/TR/xml-exc-c14n/

Digital Certificates

For distribution of owner's Public key

Irrefutably ties public key to owner

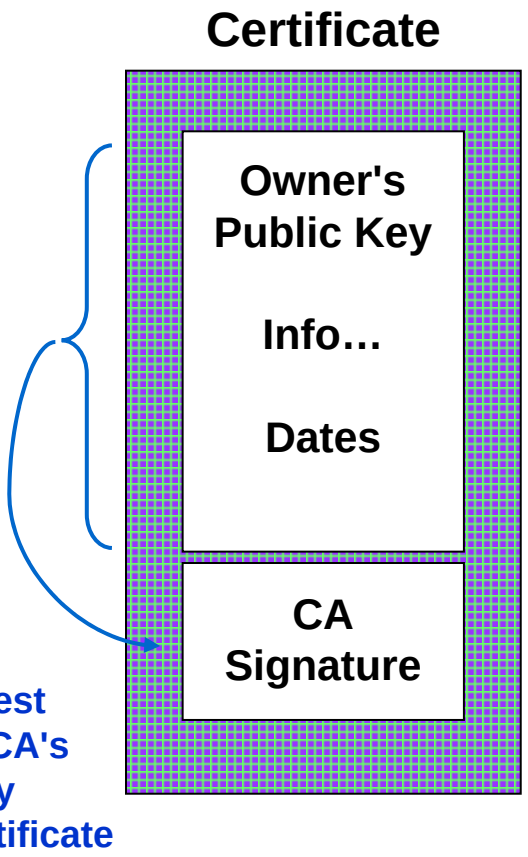
Contains

- Owner's distinguished name
- Owner's public key
- Time stamp and other information

Issued and signed by a **C**ertification **A**uthority

- Digital signing proves the certificate is valid

Common type is X.509 certificate



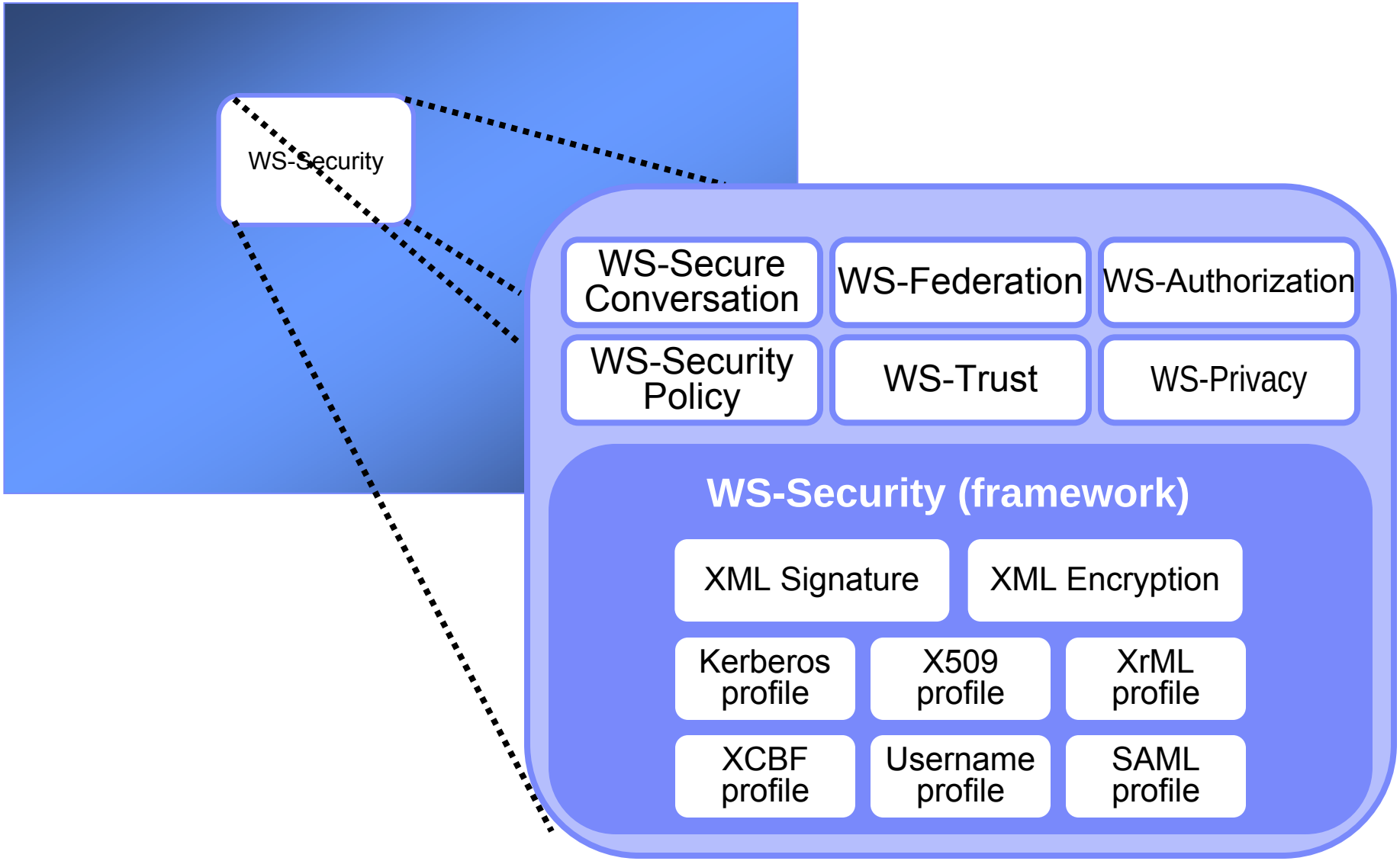
- PKI depends on Trust
  - Must trust key-issuer's authority
- For an in-depth introduction to PKI, visit: http://ibm.com/developerworks/security/library/s-pki.html

WS-Security

WS-Security defines how to use and build upon existing security technologies

As of early April 2004, WS-Security is now an OASIS standard

Security Specifications



WS-Security: SOAP Message Security

A foundational set of SOAP message extensions for building secure Web services

- Defines new elements to be used in SOAP header for message-level security

Defines the use of formerly incompatible proven and emerging security technologies:

- Kerberos, PKI, HTTPS, IPSEC, XrML
- XML Signature, XML Encryption, XKMS from W3C
- SAML, XACML from OASIS

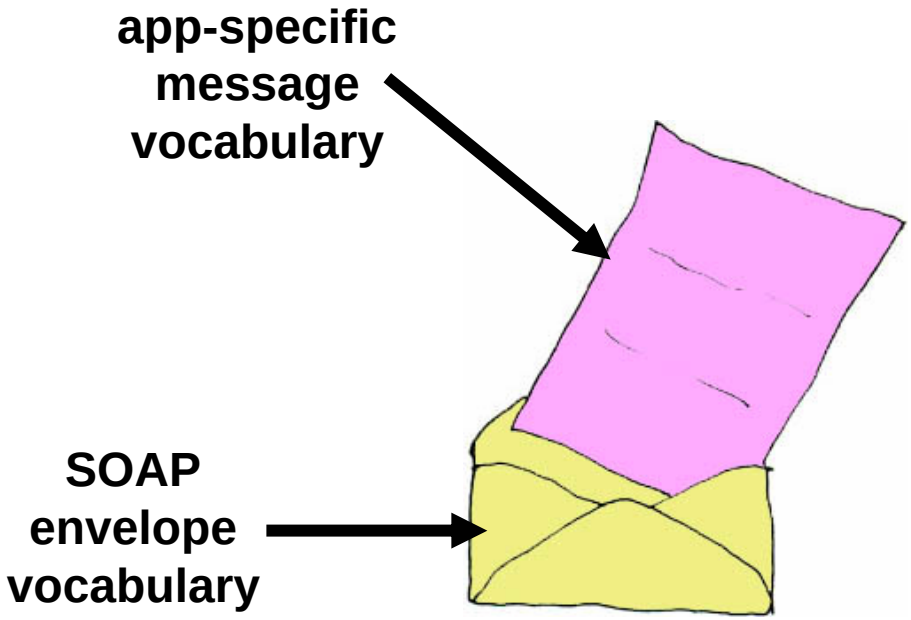
OASIS WS-Security 1.0 standard

- http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- widely supported in application servers and development tools from several vendors, including IBM, Microsoft, BEA, …

Review: SOAP Message Structure

The SOAP specification defines the "envelope“ vocabulary

- The "envelope" wraps the message itself
- The message is a different vocabulary
- A namespace prefix is used to distinguish the two parts



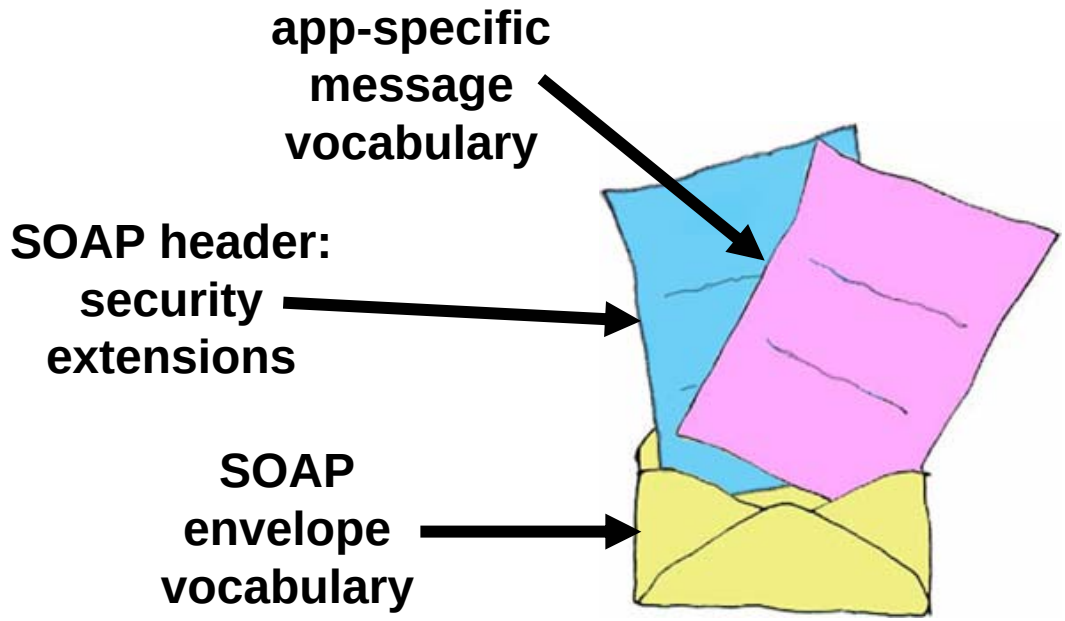
Review: SOAP Message Structure

The SOAP specification defines the “envelope” vocabulary

- The "envelope" wraps the message itself
- The message is a different vocabulary
- A namespace prefix is used to distinguish the two parts

WS-Security defines extensions to the “envelope” vocabulary

- Container for security tokens
  - Username
  - x.509 certificate
  - Kerberos ticket
  - XrML
  - XML Signature
  - SAML
- Encryption details

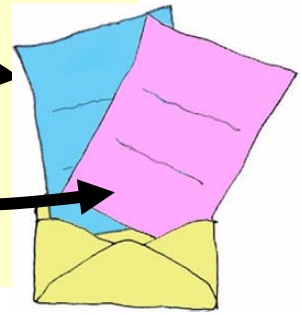


The WS-Security <Security> element

The WS-Security specification defines a vocabulary that can be used inside the SOAP envelope. **<wsse:Security>** is the "container" for security-related information.

Define and use WS-Security namespace

```
<S:Envelope
   xmlns:S="http://www.w3.org/2002/06/soap-envelope">
   <S:Header>
      <wsse:Security
         xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/07/secext">
            Security information
      </wsse:Security>
   </S:Header>
   <S:Body> App-specific content </S:Body>
</S:Envelope>
```



SOAP Envelope

Security Tokens for the <Security> Element

A **Security Token** is a collection of one or more "claims"

- A claim is a declaration made by some entity, such as name, identity, key, group, privilege, capability, etc
- "username" is an example of an unsigned security token

A **Signed Security Token** is one that is cryptographically signed by a specific authority

- An X.509 certificate is a signed security token
- A Kerberos ticket is also a signed security token

An **XML Security Token** is one that is defined with a separate XML schema, rather than simple or encrypted text

- SAML and XrML are examples
- Can be included directly in **<wsse:Security>** container

The WS-Security <**UsernameToken**> element

This element can be used to provide a user name within a <**wsse:Security**> element, for **Basic Authentication**

```
<S:Envelope
   xmlns:S="http://www.w3.org/2002/06/soap-envelope">
   <S:Header>
     <wsse:Security
        xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/07/secext">
        <wsse:UsernameToken wsu:ID="myToken">
           <wsse:Username>tigerplaid</wsse:Username>
           <wsse:Password>passw0rd</wsse:Password>
        </wsse:UsernameToken>                    Security Info
     </wsse:Security>
   </S:Header>
   <S:Body>
        App-specific content
   </S:Body>
</S:Envelope>
```

Basic Authentication Demo

We have decide that our StockTrade service should only be available to subscribers, who must supply their username and password to obtain a quote.

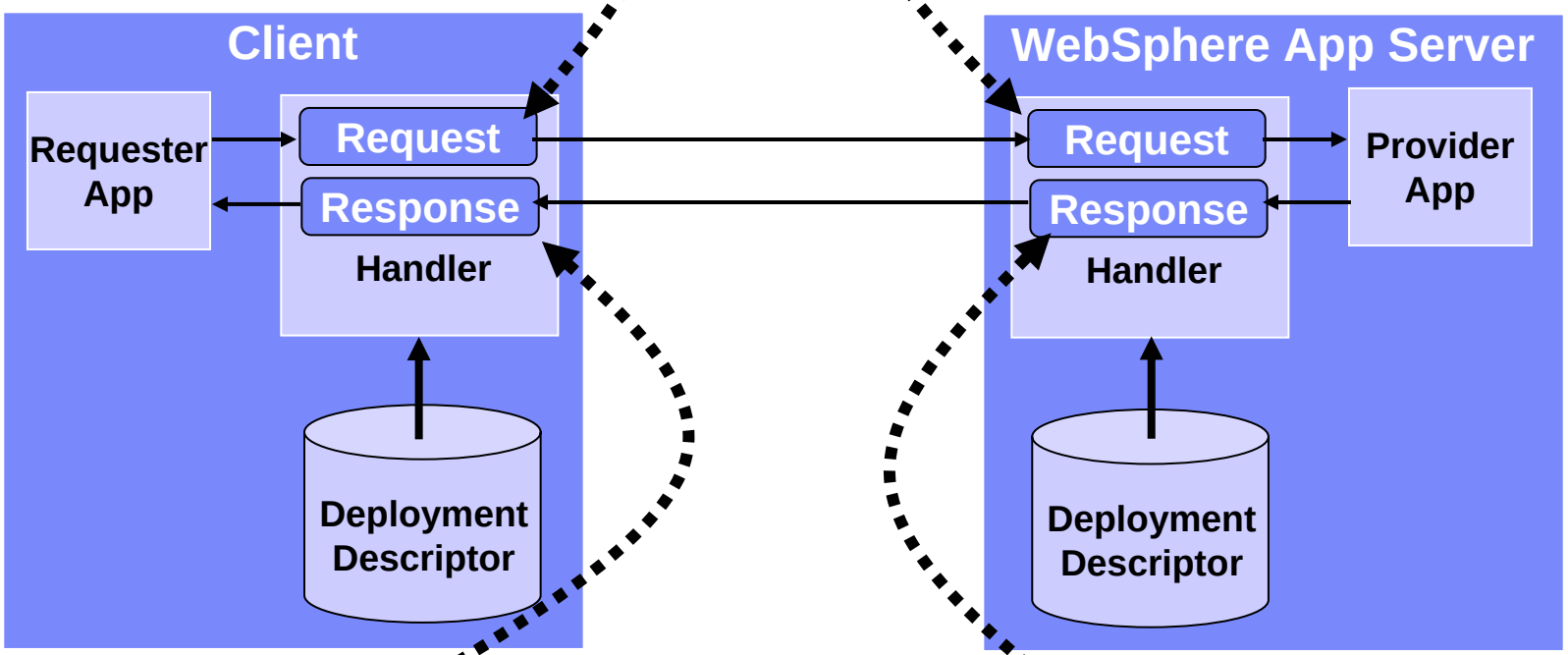
WebSphere Studio makes it easy to add WS-Security for this and other purposes.

- The Security Handler module contains the code for Web Services Security functions; the *deployment descriptor* tells it what it should do for us.
- We simply edit the *deployment descriptor* for the Service, and for the Service Requester application. We don't add or change any code in the Java source!
  - This allows a security manager to control and change all security options without interfering with developer tasks

WS-Security Implementation in WebSphere

SOAP request header construction
- Security token generation
- Digital signature generation
- Content encryption

SOAP request header processing
- Validate security tokens
- Set up security context
- Decrypt content
- Digital signature validation



SOAP response header processing
- Decrypt content
- Digital signature validation

SOAP response header construction
- Digital signature generation
- Content encryption

The WS-Security <BinarySecurityToken> element

Signed security tokens, such as a Kerberos ticket or x.509 certificate, are binary content. They must be encoded for inclusion in the wsse:Security container

```
<S:Envelope
   xmlns:S="http://www.w3.org/2002/06/soap-envelope">
   <S:Header>
      <wsse:Security
         xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/07/secext">
         <wsse:BinarySecurityToken wsu:ID="myToken"
            ValueType="wsse:Kerberosv5ST"
            EncodingType="wsse:Base64Binary>
            XIFNWZz99UUbalqIEmJZc0
         </wsse:BinarySecurityToken>          Security Info
      </wsse:Security>
   </S:Header>
   <S:Body> App-specific content </S:Body>
</S:Envelope>
```

Using XML Digital Signature with SOAP

As we have seen, XML Digital Signatures tells us how to sign arbitrary XML content

How do we use XML Signatures with SOAP messages?

- WS-Security defines new elements in the SOAP header and body to contain an XML Signature
- Standardization of these elements means that implementations from different vendors can interoperate with signatures
- WS-I Basic Security Profile specifies details to ensure that interoperability

Example : SOAP with XML Signature

```
<S:Envelope>
  <S:Header>
    <wsse:Security S:mustUnderstand="1"
       xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext">
      <wsse:BinarySecurityToken EncodingType="wsse:Base64Binary">
         MIIDQTCC4ZzO7tIgerPlaid1q ... [truncated]
      </wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      ....see XML Signature example for full content...
      </ds:Signature>
    </wsse:Security>
 </S:Header>

 <S:Body>
    <m:OrderAircraft quantity="1" type="777" config="Atlantic"
       xmlns:m="http://www.boeing.com/AircraftOrderSubmission"/>
 </S:Body>
<S:Envelope>
```

WS-Security Uses XML Encryption elements

<EncryptedData> element replaces the content being encrypted.

It contains:

- **<EncryptionMethod>** Algorithm used to encrypt the data
- **<CipherData>**
  - **<CipherValue>** Element containing the encrypted data

<EncryptedKey> element placed in security header, contains

- **<EncryptionMethod>** Algorithm used to encrypt symmetric key
- **<KeyInfo>** Identifier of key used to encrypt symmetric key
- **<CipherData>**
  - **<CipherValue>** Encrypted symmetric key value
- **<ReferenceList>** List of **<DataReference>**s to content encrypted with this symmetric key

Example: entire <body> contents encrypted

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith<Name/>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PayBalanceDue >
```

Red text is data to be encrypted
Green text is left unencrypted

```
<EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
  Type='http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml'>
  <CipherData><CipherValue>A23B4C6</CipherValue></CipherData>
</EncryptedData>
```

"PayBalanceDue" element identity is hidden in encrypted form. We can't even see what kind of transaction it is!

The real cipher would be longer than this

Example: one element and subelements encrypted

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith<Name/>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PayBalanceDue >
```

Red text is data to be encrypted
Green text is left unencrypted

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith<Name/>
  <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
    Type='http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml'>
    <CipherData><CipherValue>A23B4C6</CipherValue></CipherData>
  </EncryptedData>
</PayBalanceDue >
```

<CreditCard> group was replaced by an EncryptedData element

Example: element text (only) encrypted

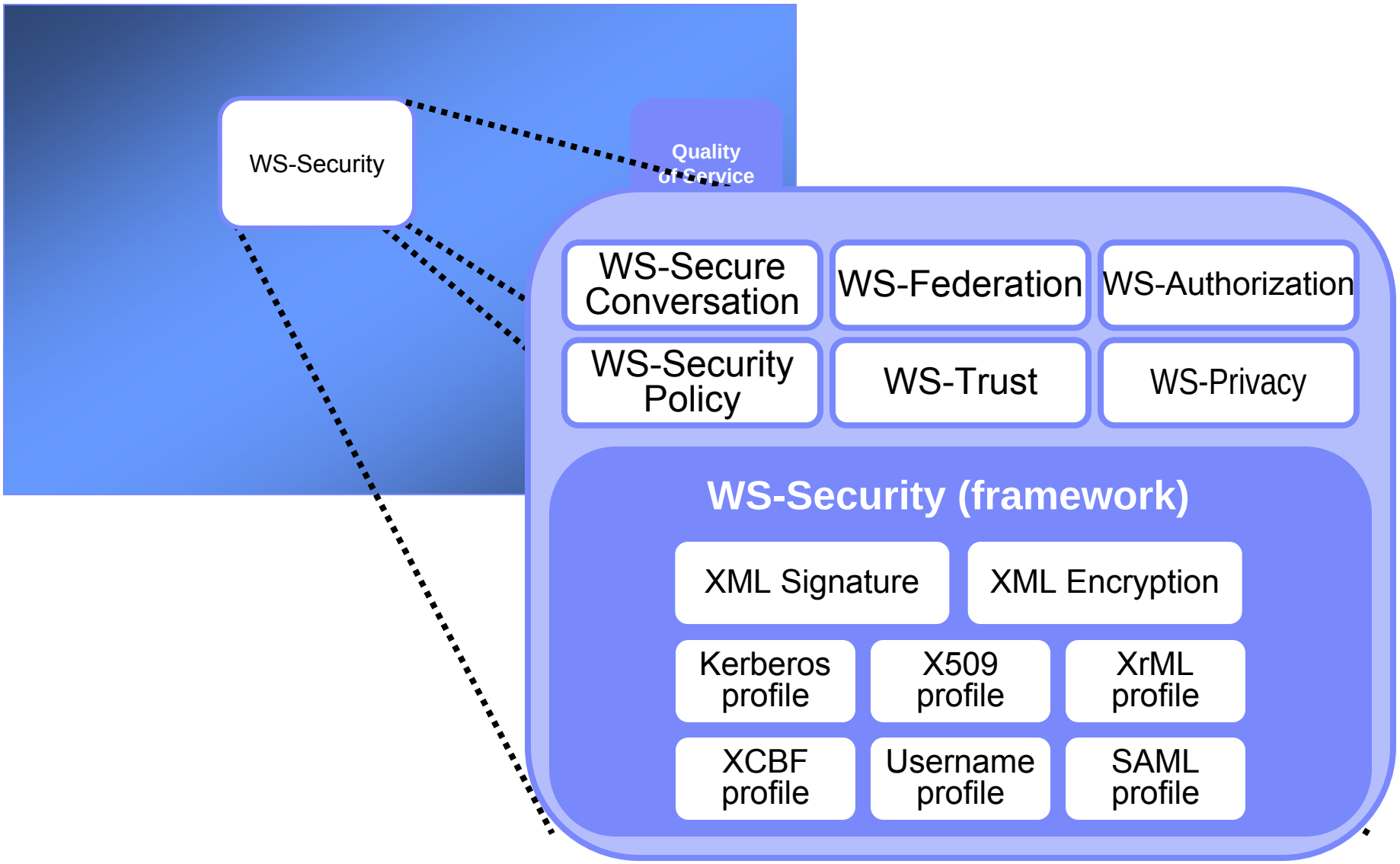
```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith<Name/>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4018 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PayBalanceDue >
```

Red text is data to be encrypted
Green text is left unencrypted

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith<Name/>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>
      <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
        Type='http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml'>
        <CipherData><CipherValue>A23B4C6</CipherValue></CipherData>
      </EncryptedData>
    </Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
</CreditCard>
</PayBalanceDue >
```

Text was replaced by an EncryptedData element

WS-Security Roadmap



What is a policy?

A ***policy*** is a set of capabilities, requirements, preferences, and general characteristics about entities in a system

The elements of a policy (***policy assertions***) can express:

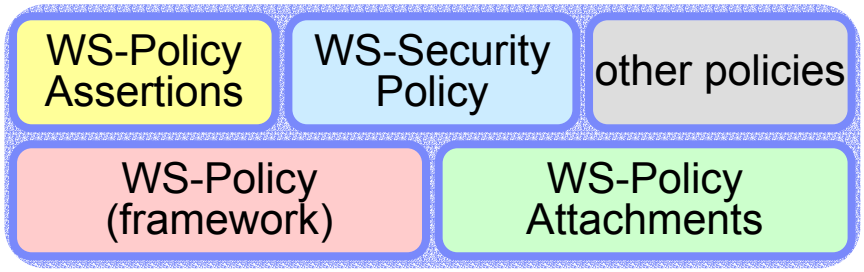
- Security requirements or capabilities
- Various Quality of Service (QoS) characteristics
- Any other kinds of policies that are required

WS-Policy defines a general purpose, extensible model and grammar ("***framework***") for describing policies in a Web services system

- Simple, declarative policies
- More complex, conditional policies

WS-Policy

http://ibm.com/developerworks/webservices/library/ws-polfram



WS-Policy defines the framework for policy definition

- The container element <Policy>
- The organizing operator elements
- The “Preference” and “Usage” concepts / attributes
- An inclusion / reuse mechanism

Other policy-related specifications::

- WS-PolicyAssertions – character encoding, national language
- WS-SecurityPolicy – security-related policy assertions
- WS-PolicyAttachment - how to attach a policy to WSDL or UDDI
- Policies for other technologies – e.g. WS-Privacy, WS-Authorization, WS-ReliableMessaging

Example Policy

Policy Expression
- XML InfoSet
- Describes combinations of assertions

Operators
- Describes semantics of combinations of assertions

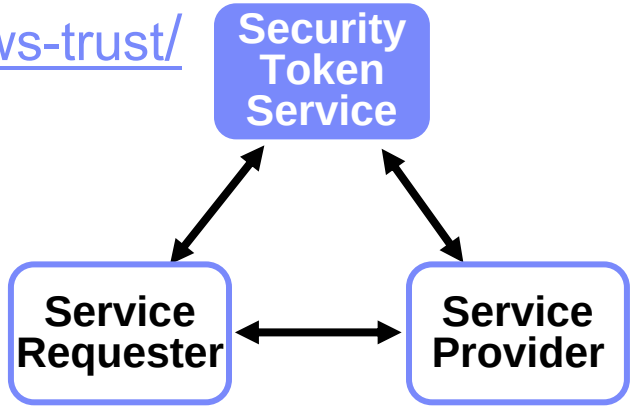
Assertions
- The leaf nodes in the policy expressions
- No core assertions defined in WS-Policy
- New assertions can be defined using namespace mechanisms

```
<Policy>
  <ExactlyOne>
    <All>
       <SecurityToken …STA…/>
       <Algorithm …AA…/>
    </All>
    <All>
       <SecurityToken …STB…/>
       <Algorithm …AB…/>
    </All>
  </ExactlyOne>
</Policy>
```

Operator **<All>** – every contained element must be applied
<ExactlyOne> – one of the contained elements must be applied
<OneOrMore> – one or more of the elements must be applied

WS-Trust

http://ibm.com/developerworks/webservices/library/ws-trust/



A model for direct and brokered trust relationships

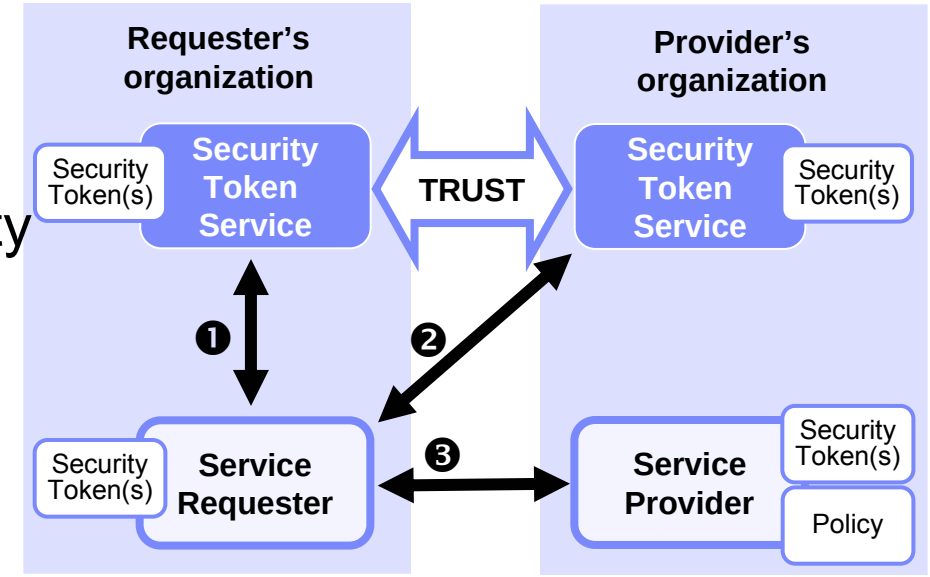
- Third parties and intermediaries
- Manage credentials across different trust domains

Defines:

- The "**security token service**"
  - A trusted authority for security tokens implemented as a Web service
- SOAP messages sent to this service for security token issuance, validation and exchange

WS-Federation

http://ibm.com/developerworks



A *federation* is a collection of security realms (e.g. partner organizations) that have established trust to share security information about users belonging to the realms:

- identification, authentication
- attributes, authorization

The WS-Federation Specification:

- builds on the WS-Trust model
- can share this data using different or like mechanisms
- defines mechanisms for the brokering of trust and for security token exchange between trust domains
- does not require local identities at target services
- optionally allows hiding of identity info and other attributes

WS-Secure Conversation

http://ibm.com/developerworks/webservices/library/ws-secon/

Establish a secure, shared security context in which to exchange multiple messages

Defines the mechanisms for

- Establishing and sharing security contexts
- Deriving session keys from security contexts

Defines 3 ways of establishing a security context

- Security context token created by a security token service
- Security context token created by one of the communicating parties and propagated with a message
- Security context token created through negotiation

SecurityContextToken Example

- SecurityContext header
- Identifies the security context using a URI
- Indicates the creation time of the security context
- Indicates the expiration time of the security context
- Holds the shared secrets of the security context
- References a shared secret of the security context

```
<wsse:SecurityContextToken wsu:Id="...">
   <wsu:Identifier>...</wsu:Identifier>
   <wsu:Created>...</wsu:Created>
   <wsu:Expires>...</wsu:Expires>
   <wsse:Keys>
      <xenc:EncryptedKey Id="...">...
      </xenc:EncryptedKey>
      <wsse:SecurityTokenReference>...
      </wsse:SecurityTokenReference>
      ...
   </wsse:Keys>
</wsse:SecurityContextToken>
```

WS-Privacy

Planned.

Privacy refers to policies of how a service provider will use information supplied by a service requester.

WS-Privacy will be a model for how users state privacy preferences, and for how Web Services state and implement privacy practices.

Preferences are specified by way of policy assertions in the WS-Policy container.

More information: see Web Services Security Roadmap at

- http://ibm.com/developerworks/webservices/library/ws-secmap/

WS-Authorization

Planned.

Will describe how access policies for a Web service are specified and managed

Will describe how claims may be specified within security tokens, and how these claims will be interpreted at the endpoint

- More information: see Web Services Security Roadmap

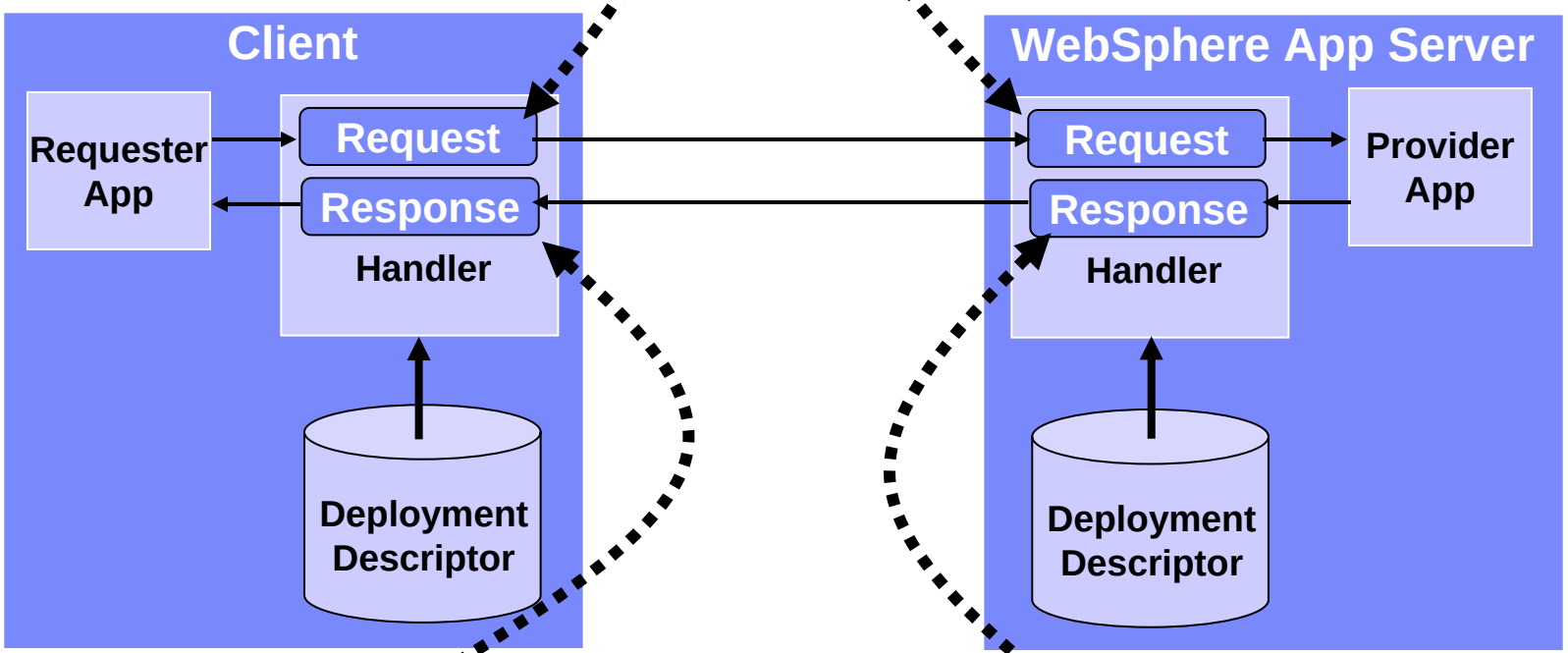
  http://ibm.com/developerworks/webservices/library/ws-secmap/

WS-Security Product support

WS-Security Implementation in WebSphere

SOAP request header construction
- Security token generation
- Digital signature generation
- Content encryption

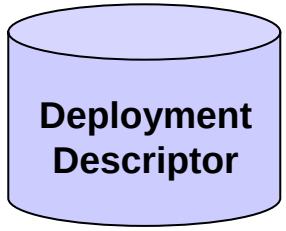
SOAP request header processing
- Validate security tokens
- Set up security context
- Decrypt content
- Digital signature validation



SOAP response header processing
- Decrypt content
- Digital signature validation

SOAP response header construction
- Digital signature generation
- Content encryption

Deployment Descriptor for Security requirements



WS-Security requirements are specified as security constraints in the Deployment Descriptor:

- Should the message be digitally signed or encrypted?
- What is the trust mode for identity assertion
- What are the security tokens to be used as the caller identity

The Security Handlers act on the specified constraints to enforce WS-Security requirements

This approach supports a separation of roles:

- Developer of Web Service provider or requester app
- Assembler or deployer of Web Service

It also makes it easy to revise security requirements, since they are specified separately from the application code.

- Microsoft .Net approach generates code to handle security; this is less flexible for dealing with changing security requirements

Resources

Resources: Security Specifications

Latest WS-Security specifications are available on http://oasis-open.org

- Incorporates errata, includes changes from working group

Other specs are available on http://ibm.com/developerworks

- Search for WS-Security to get the entire list

Whitepaper: “Web Services Security: Moving up the stack“

- http://ibm.com/developerworks/webservices/library/ws-secroad/
- Published December, 2002

Original plan for WS-Policy is described in the WS-Security Roadmap

- http://ibm.com/developerworks/webservices/library/ws-secmap/
- Published April, 2002. Differs somewhat from specifications, which are the definitive source of information

WS-I Basic Security Profile Working Draft

- http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0-2004-05-12.html

Free WebSphere SDK (WSDK)

WebSphere SDK 5.1

- On-ramp to Web services for Java programmers
- An integrated kit for creating, discovering, invoking, and testing Web services
- Embedded IBM WebSphere Application Server Express V5 included
- Full support for WS-Security, XML Encryption, XML Digital Signatures
- Includes an implementation of JDK 1.4.1
- Migration is available to production version of IBM WebSphere Application Server V5
- Versions available FREE for Windows and Linux

Available for free download from

http://ibm.com/developerworks/offers/ws-speed-start/wsdk.html

Free Tutorial: Securing Web Services

Learn more about WS-Security from this new tutorial:

- http://www-106.ibm.com/developerworks/ibm/edu/i-dw-wes-buildsecws-i.html

Discusses:

- Hands-on with WebSphere Studio Application Developer
- Basic security requirements
- WS-Security and XML Encryption, XML Signature
- Adding XML Encryption to SOAP messages
- Adding XML Digital Signature to SOAP messages
- Adding Basic Authentication to SOAP messages
- Details of security header contents in each case

ETTK - Emerging Technologies Toolkit*

Contains many technologies useful for Web services

- available on http://ibm.com/alphaworks

For Web Services Security, it features demos:

- **AXIS Digital Signature Demo**: uses Apache Axis handlers to process digital signatures without app-level code
- **AXIS Encryption Demo**: same for XML Encryption processing
- **Federated Identity Demo**: supports cross enterprise authentication within a heterogeneous federation of services and security systems
- **Security Policy demo**: client evaluates the policy assertions available for two services and determines whether the request should be digitally signed or not
- **WS-Privacy demo**: a simple web app maintains personal data in the Profile Service, and grants or denies access to that data based on authorization decisions made by the Privacy Policy
- **XKMS** key management demo

* The software formerly known as WSTK – Web Services Toolkit

Thank You!

Acknowledgements: material and ideas in this presentation come from many inside IBM, including (but not limited to) Doug Tidwell, Anthony Nadalin, Maryann Hondo, and countless others. Thanks also to Joan McCandlish for her drawing of the envelope and the papers inside.

Visit http://ibm.com/developerworks/speakers/colan for PDFs of this and other presentations on SOA, Web Services, and XML technologies.