Securing Web Services and Beyond

Anthony Nadalin
IBM Corporation
In collaboration with MaryAnn Hondo and Nataraj Nagaratnam
The Evolution to Web Services

Pre-1990s
Custom, static
B2B Integration

Early 1990s
Application integration
technologies appear

Late 1990s
Web technologies appear
e.g. HTTP, HTML, XML

2000+
Web application
technology = Web services

Business Benefit
Web Services Today

- Web services is currently an early majority market
  - Many successful deployments
  - A 1st or 2nd focus for CIOs and architects
- Web services add immediate value to a broad range of scenarios
- Base infrastructure – SOAP and WSDL – have broad adoption and tool integration
- Great progress toward enabling security, reliability, and transactions

"Web services development projects are at the top of the list of company priorities and one of the last budgets to be raided when budget cuts are made."
- Gartner survey of 111 North American Companies

InfoWorld CTO Survey

"In what areas will web services be most effective for your company?"

- B-to-B eCommerce: 70%
- Supply Chain: 33%
- CRM: 33%
- B-to-C eCommerce: 24%
- ERP: 15%
- Other: 6%
Real World Web Services
Connecting Everything

- Personal Networks
- Government Networks
- Social Networks
- Customer Relationship Networks
- Scientific Networks
- Value Chain Networks
- Business Networks
- Fulfillment
- HR
- Purchasing
WS-* Specifications Timeline

- **April 2002**: WS-Security
- **June 2002**: WS-Coordination
- **August 2002**: WS-Transaction
- **December 2002**: WS-SecurityPolicy
- **March 2003**: WS-Coordination
- **April 2003**: WS-SecureConversation
- **June 2003**: WS-Trust
- **July 2003**: WS-Policy
- **September 2003**: OASIS WS-Security V1 Last Call

- **March 2003**: WS-Policy
- **April 2003**: WS-ReliableMessaging
- **June 2003**: WS-PolicyAttachment v1.1
- **July 2003**: WS-Addressing
- **September 2003**: WS-PolicyAssertions v1.1
- **September 2003**: Reliable Message Roadmap

- **March 2003**: WS-PolicyAttachment
- **April 2003**: WS-Addressing
- **June 2003**: WS-PolicyAttachment v1.1
- **July 2003**: WS-PolicyAssertions v1.1
- **September 2003**: Reliable Message Roadmap
Messaging

Addressing

SOAP Messaging
WS-Addressing

- Describes transport-neutral mechanisms to address Web Services and messages
- Identification of Web Service end points
- End point reference: URI + application-specific information
- End-to-end identification in messages
Policy Framework

Policy Attachment

Policy Assertions

Policy

WSDL
WS-Policy

- Flexible and extensible grammar for Web Services to communicate requirements, preferences and capabilities
- Declarative and conditional assertions
  - Authentication scheme
  - Transport protocol
  - Security policy
  - QoS characteristics
  - ...

...
WS-Policy

- Policy assertions have usage attributes
  - Required, Rejected, Optional, Observed, Ignored

- Policy operators
  - All – all of its child elements are satisfied
  - ExactlyOnce – exactly one of its child elements is satisfied
  - OneOrMore – at least one of its child elements is satisfied

- PolicyReference element for inclusion
WS-PolicyAssertions

- Describes general policy assertions that can be affiliated with a message
  - TextEncoding assertion
  - Language assertion
  - SpecVersion assertion
  - MessagePredicate assertion
- Supported by both the Web Service or by the client
WS-PolicyAttachment

- Provides a standard mechanism for attaching the requirement and capability statements to Web Services
  - How to associate policies with specific instances of WSDL services
  - How to reference policies from WSDL definitions
  - How to associate policies with UDDI entities
WS-Security

- Defines a framework for building security protocols
  - Integrity
  - Confidentiality
  - Propagation of security tokens
- Framework designed for end-to-end security of SOAP messages
  - From initial sender, through 0-n intermediaries to ultimate receiver
WS-Security

- Leverages existing XML security specs
  - XMLDSIG for integrity
  - XMLENC for confidentiality
- Provides constructs for transmitting security tokens
  - Supports XML and binary tokens
What Are Security Tokens?

- Examples include
  - Username token
  - X509 Certificate
  - Kerberos ticket
  - XrML license
  - SAML assertion

- Represent claims about
  - Identity
  - Capabilities
  - Privileges
Security Token Example

- Message claims to be from Alice
  - Specified using Alice's X509 certificate
- Proof is based on Alice's private key
  - Signing part of the message with her private key proves that she knows the key and is therefore Alice
  - Specifically, that the signed parts are from Alice
Protecting Messages

- Parts of a message can be
  - signed for integrity
  - encrypted for confidentiality
- Underlying technologies extensible
  - Encryption
  - Digest
  - Signature
  - Canonicalization
  - Transforms
What does a secure Web Services message transmission look like? (SOAP example)

**Soap Envelope**
- Soap Header
- Message Header and Routing
- Security Content
  - Security Token
  - Signature
    - Actual signed content
- Message Body
Security Needs for Web Services

- A web service is a series of transactions
  - Service Consumer to Service Registry to obtain location of desired services
  - One or more requests from the Service Consumer to one or more Service Provider(s)
  - One or more responses from the Service Provider(s) back to the Service Consumer
- The security requirements are the same as for any other transaction
  - Confidentiality
  - Integrity
  - Non-repudiation
- Protections can include:
  - Protecting the transmission path (internal network, dedicated circuit)
  - Protecting the transmission itself (SSL, VPN, encrypted circuit)
  - Protecting the payload during transmission (encrypted file, *i.e.*, PGP, S/MIME)
- Without specific protections, the transmissions are clear text, as they are based on HTTP
Web Services Security Standard

- **WS-Security** describes enhancements to SOAP messaging to provide *quality of protection* through message integrity, message confidentiality, and single message authentication.

- **WS-Security** describes how to encode binary security tokens. Specifically, the specification describes how to encode X.509 certificates and Kerberos tickets as well as how to include opaque encrypted keys.

- **WS-Security** does not ensure security nor does it provide a complete security solution. WS-Security is a building block that is used in conjunction with other Web service and application-specific protocols to accommodate a wide variety of security models and encryption technologies. Implementing WS-Security does not mean that an application cannot be attacked or that the security cannot be compromised.

- **WS-Security** is flexible and is designed to be used as the basis for the construction of a wide variety of security models including PKI, Kerberos, and SSL. Specifically WS-Security provides support for multiple security tokens, multiple trust domains, multiple signature formats, and multiple encryption technologies.
WS-Security

- WS-Security does provide:
  - Message level security
  - "Improved SSL"
  - Security at lower/network layer
  - Transmission security
  - Message authentication
  - Message confidentiality
  - Message integrity

- WS-Security does NOT provide:
  - Application level security
  - Enterprise security
  - Authentication mechanisms
  - Authorization security
  - Intrusion detection
  - Identity management
  - Security Architecture
  - Network Security
  - Anti-Virus protection
Overall Security Requirements

In any architectural solution, the following security requirements must be addressed, with no exceptions when it comes to Web services:

- **Identification:** The party accessing the resource is able to identify itself to the system.
- **Authentication:** There exists a procedure to verify the identity of the accessing party.
- **Authorization:** There exists a set of transactions the authenticated party is allowed to perform.
- **Integrity:** The information is not changed on its way.
- **Confidentiality:** Nobody is able to read the information on its way.
- **Auditing:** All transactions are recorded so that problems can be analyzed after the fact.
- **Non-repudiation:** Both parties are able to provide legal proof to a third party that the sender did send the information, and the receiver received the identical information.
WS Security

Terminology:

- **Claim** – A *claim* is a statement that a client makes (e.g. name, identity, key, group, privilege, capability, etc.).
- **Security Token** – A *security token* represents a collection of claims.
- **Signed Security Token** – A *signed security token* is a security token that is asserted and cryptographically endorsed by a specific authority (e.g. an X.509 certificate or a Kerberos ticket).
- **Proof-of-Possession** – The *proof-of-possession* information is data that is used in a proof process to demonstrate the sender's knowledge of information that SHOULD only be known to the claiming sender of a security token.
- **Integrity** – *Integrity* is the process by which it is guaranteed that information is not modified in transit.
- **Confidentiality** – *Confidentiality* is the process by which data is protected such that only authorized actors or security token owners can view the data.
- **Digest** – A *digest* is a cryptographic checksum of an octet stream.
- **Signature** – A *signature* is a cryptographic binding of a proof-of-possession and a digest. This covers both symmetric key-based and public key-based signatures. Consequently, non-repudiation is not always achieved.
- **Attachment** – An *attachment* is a generic term referring to additional data that travels with a SOAP message, but is not part of the SOAP Envelope.
WS Security Capabilities

Summary

- **Message Security Model**
  - Security Tokens MAY be bound to messages

- **Message Protection**
  - Message Integrity – attained by using XML Signatures with Security Tokens
  - Message Confidentiality – attained by using XML Encryption with Security Tokens

- **WS Security Standard allows:**
  - Encryption/Signing of:
    - Body
    - Body Elements
    - Header
    - Attachments
WS Security Message Example

Message example with a username security token (1 of 3):

(001) <?xml version="1.0" encoding="utf-8"?>
(002)  <S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
                xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
(003)   <S:Header>
(004)     <m:path xmlns:m="http://schemas.xmlsoap.org/rp/>
(005)     <m:action>http://fabrikam123.com/getQuote</m:action>
(006)     <m:to>http://fabrikam123.com/stocks</m:to>
(007)     <m:id>uuid:84b9f5d0-33fb-4a81-b02b-5b760641c1d6</m:id>
(008)   </m:path>

First two lines start SOAP message
Lines 004 to 008 define how to route this message
WS Security Message Example

Message example with a username security token (2 of 3):

(009)  

_LINE 009: Start of Security header_

(010)  

_LINES 010 to 012 specify the security token_

(011)  

(012)  

(013)  

(014)  

(015)  

_LINE 015: Canonicalization Method Algorithm_

(016)  

_LINE 016: Signature Method Algorithm_

(017)  

_LINE 017: Reference URI_

(018)  

_LINE 018: Digest Method Algorithm_

(019)  

_LINE 019: Digest Value_

(020)  

_LINE 020: Reference_
WS Security Message Example

Message example with a username security token (3 of 3):

(021)   </ds:SignedInfo>
(022)   <ds:SignatureValue>DJbchm5gK... </ds:SignatureValue>
(023)   <ds:KeyInfo>
(024)     <wsse:SecurityTokenReference>
(025)       <wsse:Reference URI="#MyID"/>
(026)     </wsse:SecurityTokenReference>
(027)   </ds:KeyInfo>
(028)   </ds:Signature>
(029)   </wsse:Security>
(030)   </S:Header>
(031)   <S:Body Id="MsgBody">
(032)     <tru:StockSymbol xmlns:tru="http://fabrikam123.com/payloads">
             QQQ
     </tru:StockSymbol>
(033)   </S:Body>
(034)   </S:Envelope>

Lines 031 to 033 contain the body of the SOAP message
WS-Security

Sender  Intermediary  Intermediary  Receiver
**WS-Trust**

- Defines how to broker trust relationships
  - Some trust relationship has to exist *a priori*
- Defines how to exchange security tokens
- Defined as an interface specification for a Security Token Service
- Anyone can issue tokens (be a Security Token Service)
Getting Tokens

- A RequestSecurityToken message is sent to the trust service
- It responds with a RequestSecurityTokenResponse
  - Contains required security token and associated details (*e.g.* proof)
Example

- I want to have secure communication with you
- I ask the trust service for a token to allow me to talk to you
- The trust service sends two copies of a secret key
  - One encrypted for me (proof token)
  - One encrypted for you (requested token)
Example

1. U/P
   - T₁
   - P₁

2. T₁
   - T₂
   - P₂

3. T₂

Security Token
Proof token
Challenges

Request Token

Issue Challenge

Respond to Challenge

Issue Token
Other Token Characteristics

- Requester can specify various required characteristics of the security token
  - Key type, size
  - Delegation constraints
  - ...

- Trust service can then indicate those characteristics in the response
  - May indicate anything it thinks important
WS-SecureConversation

- WS-Security provides for single message security
- Nodes will often want to exchange more than one message
  - Specifying new symmetric keys for each message is tedious, verbose, and inefficient
- WS-SecureConversation defines mechanisms to address this
WS-SecureConversation

- Participants establish a shared context
  - Context contains keys/secrets and other information
  - Can be stateless (state embedded in security context token)

- Context established multiple ways
  - Using token exchange
  - Having one party create the context
  - Through negotiation
Persisted Context
Farm Context
WS-SecurityPolicy

- A set of policy assertions related to concepts defined by other WS-Sec* specs
- Allows participants to specify
  - Token types
  - Whether integrity and/or confidentiality are required
  - Algorithms for the above
  - Which message parts need signing/encrypting
WS-Federation

- "Single Sign-On" access across trust domains using identities from the different domains
- WS-Federation defines a model for this building on the WS-* security specifications:
  - Model for trust
  - Sign out messages
  - Attribute service
  - Pseudonym service
Secure, Reliable, Transacted Web Services

- BPEL4WS
- Service Composition
- Security
- Reliable Messaging
- Transactions
- Composable Service Assurances
- XSD, WSDL, UDDI, Policy, MetadataExchange
- Description
- XML, SOAP, Addressing
- Messaging
- HTTP, HTTPS, SMTP
- Transports

From joint IBM/MSFT WS Whitepaper at 
Reliable Messaging

- Policy
- Security
- SOAP Messaging
WS-ReliableMessaging

- End-to-end delivery of messages with specific quality-of-service characteristics among two parties
  - Identification of sequences of messages
  - Specification of delivery assurances
    - At most once, Exactly once, and In-order delivery
  - From initial sender, through 0-n intermediaries to ultimate receiver
- No restriction on the number of in-flight messages
- Transport-independent
- Integrated with WS-* security mechanisms
WS-ReliableMessaging

Send message #1
Send message #2
...Send message #1
Acknowledge #1-2
Send message #3
...

1-4 1-2

1-3 1-2
Importance of Composition

- Everything works in combination
  - Ex: Transaction context works over a reliable connection
  - Ex: Participants use WS-Security to secure transactions (for all types participants)

- Not "reinventing the wheel" for every stack
  - Code reuse, lower costs, faster time to market
  - Ex: all resources named using WS-Addressing

- The overall system is more stable
  - Changes don't percolate up the stack
  - Ex: By using WS-Security, Federation supports all tokens, including future ones
Composable Headers

<Addressing>

<wsa:ReplyTo>
</wsa:ReplyTo>
<wsa:To>http://fabrikam123.com/Traffic</wsa:To>
<wssec:Security>
  <wssec:BinarySecurityToken
    ValueType="wssec:X509v3"
    EncodingType="wssec:Base64Binary">
    dWJzY3JpYmVyLVBlc…..eFw0wMTEwMTAwMD
  </wssec:BinarySecurityToken>
</wssec:Security>
<wsrm:Sequence>
  <wsu:Identifier>http://fabrikam123.com/seq1234</wsu:Identifier>
  <wsrm:MessageNumber>10</wsrm:MessageNumber>
</wsrm:Sequence>
</S:Header>

<app:TrafficStatus
  xmlns:app="http://highwaymon.org/payloads">
  <road>520W</road><speed>3MPH</speed>
</app:TrafficStatus>
</S:Body>
</S:Envelope>

<Security>
</Security>

<Reliability>
</Reliability>
OASIS Web Services Security
Major Milestones

- Authoring of WSS documents (1Q 2002)
- Public demo of WSS interop between IBM & MS at XML One Boston (August 27th)
- OASIS Standard March 31st

- TC Charter July 9th
- Public demo of WSS interop between IBM & MS at XML One Boston (August 27th)
- OASIS Standard March 31st

- 19 Months

- 2002
  - WS-Security 1.0 posted to the web along with security roadmap April 11th

- 2003
  - WSS TC first meeting, Sept 5th
  - Public review Sept 19th – Oct 19th

- 2004
  - OASIS member review Feb 15th – March 15th
  - Ballot March 16th – March 31st
These 18 companies are listed as endorsers in the initial TC charter.
Input Documents

The following documents were submitted to the TC (either at the first meeting or during following 12 months):

- WS-Security specification (IBM/MS/VRSN)
- WS-Security Addendum (IBM/MS/VRSN)
- WS-Security-XML-Tokens (IBM/MS/VRSN)
- Draft SSTC-WS-Sec-Profile-03 (SSTC – “SAML TC”)
  - This was driven by Sun and RSA
- XML Schema files (IBM/MS/VRSN)
- A “minimal” use profile for mobile/small devices (IBM)
- An XCBF profile (biometric security)
Planned Output Documents

The TC currently is working on this list of documents. Only the green list are in the WSS TC “1.0” deliverable (March 2004).

- Web Services Security: SOAP Message Security
- Web Services Security: Username Token Profile
- Web Services Security: X.509 Certificate Token Profile
- XML Schema files
- Web Services Security: Kerberos Token Profile
- Web Services Security: XrML Token Profile
- Web Services Security: SAML Token Profile
- Web Services Security: Minimal Profile
- Web Services Security: XCBF Token Profile
TC Interoperability Testing

Events

- The TC conducted two interoperability test events
  - These were private (TC only) events
  - No press or public discussion except to say it had taken place
- The first event was done at a F2F
  - About 12 companies took part
  - The TC agreed to a set of scenarios for the tests
  - In general there was a high level of success
  - Some did better than others
- The tests did not yield major spec changes
  - Mostly just clarification wording resulted
- Most of the failures were due to misinterpretation of related specs
  - Mainly XML DSIG and XML Encryption
- The second event was done over the internet
  - Added a few scenarios
  - Allowed other companies to catch up
  - Again no major spec changes resulted
  - Note: Most of the major vendors took part.
IBM WebSphere Support

- **WAS 5.0**
  - Supported WS-Security input spec as a technology preview

- **WAS 5.02**
  - Supported the first WSS TC committee draft as a partial implementation

- **WAS 5.1**
  - Increased support for the first WSS TC committee draft

- **WAS 6.0**
  - Will support full OASIS WSS TC Standard
  - Major updates
    - Timestamp processing rules
    - QNAME to URI change
    - OASIS Namespace
Work Yet to Do

- The TC now has a set of additional token profiles to complete.
  - See slide on deliverables

- An errata document has to be prepared for V1.0

- The TC does not yet have an established plan for a V2.0 effort

- Other work is TBD
Companies Represented on WSS TC (3/26/2004)

- AmberPoint
- Argonne National Lab.
- BEA Systems
- Baltimore Technologies
- Booz Allen Hamilton (p)
- CommerceOne
- Computer Associates
- ContentGuard
- Documentum
- Entregrity
- Entrust
- Fujitso
- GeoTrust
- Hewlett-Packard
- Hitachi
- IBM
- Lockheed Martin (p)
- Microsoft
- Netegrity
- Nokia
- Nortel Networks
- Novell
- Oblix
- OpenNetwork
- Oracle
- Reactivity
- RSA Security
- SAP
- Sarvega
- SeeBeyond Technology
- Sun Microsystems
- Systinet
- U.S. Dept of the Navy
- Verisign

(p) = Prospective member (not yet a voting member)
References (1 of 4)

- OASIS WSS TC Homepage
- Web Services Security: SOAP Message Security
- Web Services Security: Username Token Profile
- Web Services Security: X.509 Certificate Token Profile
- Schema Files
References (2 of 4)

- OASIS WSS TC Call for participation & Original Charter

- OASIS WSS TC Revised Charter – after first TC meeting

- OASIS Announcement of public review phase for WS-Security

- OASIS Announcement of WSS voting as a 1.0 standard

- Original DeveloperWorks posting of WS-Security, Roadmap & Addendum

- WS-Security License from IBM

- WS-Security License from Microsoft
References (3 of 4)

- OASIS WSS TC Disposition of public review/comments

- OASIS WSS TC Notes sent to OASIS at submission time

- Statements of implementation
References (4 of 4)

- OASIS WSS TC Public review comments archive
- OASIS WSS TC Latest “issues list” as of 3/23/2004
Practical Application of Security with Web Services
Web Services Technology

Adoption Barriers

- Technology Maturity
  - Interoperability
    - Between Platforms and Instances
  - Security
    - Multi-Domain and End to End
  - Usability
    - Design, Development, Deployment and Management
  - Performance
    - Comparable to non-XML Based Protocols
End to End Security Requirements

- Flexibility of Standards
  - Maximize Usage Pattern Coverage
- Extensibility of Standards
  - Leverage Enterprise Security Infrastructure
- Composability
  - With Other Messaging Services
- Application Security Analysis
  - Domain Specific Security Profiles
Web Service Technology

- Has Many Usage Patterns:
  - One Way
  - Request / Response
  - Intermediary
  - Composed
  - Business Process
One Way
Request / Response

- SOAP Client
- Request
- Response
- SOAP Server
Intermediary
Composed
Business Process

SOAP Server

SOAP Client

Message

Message

Message

Message
Web Services Message Composition

- Messages are composed of elements
- Elements may be signed
- Signatures target Roles/Actors
- Elements may be referenced by multiple signatures
- Messages may be composed of signed elements from other messages
- Origin authentication property of elements must survive recomposition
Security Technology

- Overview of available mechanisms for:
  - authentication
  - integrity
  - confidentiality mechanisms

- Including:
  - IPSEC and VPNs
  - SSL/TLS
  - HTTP Authentication
  - Web Services Secure Conversation (WS-SecureConversation)
  - PKCS7
  - XML Signature
  - XML Encryption
Mechanism Selection

- Criteria to determine appropriate mechanism including:
  - When [not] to use?
  - What [not] to protect?
  - How [not] to protect?
  - Comparative:
    - Security
    - Properties
    - Performance
    - Interoperability
    - Manageability

- How to:
  - Combine
    - Layering
    - Sequencing
Criteria for Selecting Security Mechanisms

- Interoperability
- Performance
- User/Origin authentication (persistent)
- Integrity (persistent)
- Confidentiality
- Accountability
- Manageability/Ease of Use
- End to End Security
## Security Technology

<table>
<thead>
<tr>
<th></th>
<th>Network</th>
<th>Transport</th>
<th>Message</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Authentication</strong></td>
<td>IPSec (address authentication only)</td>
<td>SSL/TLS w/client certs, HTTP Basic Auth</td>
<td>Web Services Security (WSS)</td>
<td>XML Digital Signature</td>
</tr>
<tr>
<td><strong>Confidentiality</strong></td>
<td>VPN, IPSec</td>
<td>SSL/TLS</td>
<td>WSS</td>
<td>XML Encryption</td>
</tr>
<tr>
<td><strong>Integrity</strong></td>
<td>VPN, IPSec</td>
<td>SSL/TLS</td>
<td>WSS</td>
<td>XML Digital Signature</td>
</tr>
</tbody>
</table>
Benefits of Transport Level Security – SSL/TLS

- HTTPS transport can be used to provide a very fast and secure transport for web services.
- Provides authentication through either HTTP Basic or Client certificates (X.509).
- Provides integrity between client and HTTP server by using asymmetric key cryptography to establish authenticity of server and client and to securely share a secret key.
- Provides confidentiality between client and HTTP server through efficient shared key cryptography.
- Has good support for a broad array of hardware accelerators.
- Is mature and similarly implemented by most vendors and thus, subject to few interoperability problems.
When to Use Transport Level Security to Secure Web Services

- High transaction volume
  - fast and scales well
- No intermediary processing (filtering or some form of content based routing)
- Interoperability issues associated with the emerging WSS implementations
  - SSL is mature and SSL implementations from different vendors interoperate well
- Secure attachments to web services
  - SSL encrypts all of the transport level packets, the web service headers, body and attachments are all secured
Benefits of Using Web Services Security

- End-to-end security through any number of intermediaries
- Integrity and Confidentiality of selective elements in message header and/or body
  - Parts of message can be encrypted for a particular SOAP node
  - XML Signatures can be applied across the whole SOAP Body, and/or the userNameToken, and/or a generated TimeStamp to provide integrity or prove the possession of the private key.
- Profiles for interoperable security tokens
- Extensibility for custom security tokens
- Propagation of security context
- Foundation for Web Services Security Roadmap
  - WS-Policy, WS-Trust, WS-SecurityPolicy, WS-Federation, WS-SecureConversation, ...
When to Use Web Services

Security

- Intermediaries – encrypt some parts of the message whilst leaving other parts in the clear.
- Message origin authentication – who/what sent the message
- Persistent security for messages – the message integrity and confidentiality persist beyond a transport connection.
- Future development – to enable federation, secure sessions, and policy based security for web services utilizing Web Services Roadmap.
Combining: WSS for Authentication & SSL for the Rest

- **Benefits**
  - WSS provides a flexible mechanism for passing around a security context
  - Support for multiple tokens (user, server)
  - Propagation of security context
  - Mixes well with SSL to provide the authentication while delegating integrity and confidentiality to the transport
  - SSL performance for encryption faster than WSS

- **When to Use**
  - When the application invoking the web service is a middle tier and wants to propagate the user’s identity not some generic “server” identity
  - When you want to propagate context using your own token format or a token you can create and manipulate using third party libraries
Combining Encryption and Signatures

- Incorrectly combining XML Encryption and XML Signature including:
  - Assumption of link between signer and encrypter
  - Guessing attacks
    - facilitated by plain text digest values over low entropy data

- Correctly combining XML Encryption and XML Signature including:
  - Advantages
  - Techniques
Replay and Redirection

- Mechanisms to protect against replay or redirection of data including:
  - When [not] to use?
  - What [not] to protect?
  - How [not] to protect?
  - Issues related to timestamps/clock synchronization
  - Use of nonce values
  - Specification of intended endpoints
Replay and Redirection – Mechanisms in WSS

- **Nonce (wsse:Nonce)**
  - Must be unique – typically a random value
  - Server must keep a cache of received nonces
  - A message could be replayed to a different SOAP node (nonce caches should be shared by all SOAP nodes that can be authorized with the same passwords)
  - Resending a message requires changing the nonce.
  - Must be integrity protected and cryptographically bound to message elements
  - Only defined for UsernameToken

- **Timestamps**
  - "Created" element in Username Token (wsse:UsernameToken/wsu:Created)
  - Timestamp element in SOAP header (wsu:Timestamp)
  - Determines freshness of message
  - Relies on synchronized clocks
  - Must be integrity protected and cryptographically bound to message elements
Replay and Redirection –
Mechanisms in WSS

- **Message Cache**
  - Recipient caches messages (requires lots of storage)
  - Recipient caches unique digest of message
  - Timestamps can be used to check for freshness – old messages discarded

- **WS-ReliableMessaging**
  - End-to-end delivery of messages with specific quality-of-service characteristics
    - Identification of sequences of messages
    - Specification of delivery assurances
      - At most once, Exactly once, and In-order delivery
      - From initial sender, through 0-n intermediaries to ultimate receiver
  - No restriction on the number of in-flight messages
  - Transport-independent
  - Integrated with WS-* security mechanisms
### Replay and Redirection – Attacks and Countermeasures

<table>
<thead>
<tr>
<th>Attacks</th>
<th>Countermeasures</th>
<th>Caveats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replay Username Token</td>
<td>Use Nonce and Created elements, Sign token</td>
<td>Unique Nonce, Clock Synchronization, Signing key/token needed</td>
</tr>
<tr>
<td>Replay Message w/ Username Token</td>
<td>Use Nonce and Created elements, Bind token to message through signatures</td>
<td>Unique Nonce, Clock Synchronization, Signing key/token needed</td>
</tr>
<tr>
<td>Replay Message w/ X509 token</td>
<td>Use timestamp, Bind timestamp to message through signatures</td>
<td>Clock Synchronization, Timestamp might not be unique enough</td>
</tr>
<tr>
<td>Replay Message w/ any token type</td>
<td>Combine with WS-ReliableMessaging</td>
<td>Not yet a standard</td>
</tr>
</tbody>
</table>
Redirection

- **WS-Addressing**
  - Describes transport-neutral mechanisms to address Web Services and messages
  - Identification of Web Service end points
  - End point reference: URI + application-specific information
  - End-to-end identification in messages

- **WS-Address plus WS-Security**
  - Integrity protect address information and cryptographically bind to message
    - `<wsa:To>`, `<wsa:From>`, `<wsa:ReplyTo>`, `<wsa:FaultTo>`,...
Identity, Authentication and Authorization

- Typical (J2EE-based) Authentication and Authorization maps well to

- Mechanisms to specify identity:
  - Authenticated Platform/Access Point
    - [IBC, VPN, WEP/LEAP, DNS]
  - Authenticated Endpoint
    - [SSL Certificate, HTTP Authentication]
  - Authenticated Message
    - Security Tokens
      - Security Token Reference Usage attributes
      - Web Services Security Policy

- J2EE Security Mapping
Trust and Validation

- Trust and validation mechanisms including:
  - CertPath
  - PKIPath/PKCS7
  - CRLs
  - OCSP
  - XKMS
  - Web Services Trust
  - Web Services Federation
Data Validation

- Data validation mechanisms including:
  - DTDs
  - XML Schema
  - When [not] to use?
  - What [not] to validate?
  - How [not] to validate?
Technology Availability

- Roadmap for security technology availability including:
  - WebSphere Application Server 5.0.2
  - WebSphere Application Server 5.1
  - WebSphere Application Server 6.0
  - WebSphere Application Server 6.1
  - Enterprise Services Bus
  - Web Services Gateway
  - Tivoli Access Manager/Federated Identity Manager
Technology Availability (1/3)

- Roadmap for security technology availability including:
  - WebSphere Application Server 5.0.2
    - OASIS draft 13 specification and Username token profile draft 2
    - Digital Signature: body, timestamp and security token
    - Encryption: body and username token
    - Token: username, X.509 binary security token, LTPA, Identity Assertion and pluggable token
Technology Availability (2/3)

- WebSphere Application Server 5.1
  - OASIS draft 13 specification and Username token profile draft 2
  - Nonce support in username token
- WebSphere Application Server 6.0
  - OASIS 1.0 specification, Username token profile 1.0 and X.509 Binary security token profile 1.0
    - Sign and encrypt any element
    - Pluggable Signing / Encryption algorithms
    - Pluggable Token – enhanced to support multiple tokens
    - Signing / encrypting with custom tokens.
    - Pluggable Key Retrieval Methods – abstraction for locating a key for signature or encryption
    - Order of signature or encryption is being performed
    - Nonce / timestamp support in Signature and Encryption
    - Enveloped Digital Signature for SOAP Envelope, All SOAP Headers, Security Header (wsse:Security)
    - WS-SecurityKerberos as Tech Preview (implements using plug points)
  - .NET or other vendors interoperability (scenario based)
  - WS-I Sample Application
  - Interim performance enhancement
  - Pluggable architecture as foundation for WS-SecureConversation and WS-Trust implementation
Technology Availability (3/3)

- **WebSphere Application Server 7.0**
  - Major performance enhancement
  - WS-Security Kerberos profile
  - WS-I Basic Security profile
  - WS-SecureConversation
  - Usability Improvement

- **Enterprise Services Bus**
  - Based on WebSphere Web services security support

- **Web Services Gateway**
  - Based on WebSphere Web services security support

- **Tivoli Access Manager/Federated Identity Manager**
  - WS-Federation
  - WS-Trust
  - Liberty SAML
Implementation and Deployment Models

Including:

➢ Programmable
  - JSR-105 XML Digital Signature APIs
  - JSR-106 XML Digital Encryption APIs
  - JSR-183 Web Services Message Security APIs

➢ Configurable
  - JSR-109 Implementing Enterprise Web Services

➢ Policy-Driven
  - WS-SecurityPolicy
Implementation and Deployment Models (1/2)

- Including:
  - Programmable (future)
    - JSR-105 XML Digital Signature APIs (final draft review)
    - JSR-106 XML Digital Encryption APIs (draft)
    - JSR-183 Web Services Message Security APIs (early stage)
  - Configurable (current)
    - Deployment model is an extension of JSR-109 model
    - WSAD, AST and Admin Console
  - Policy-Driven (future)
    - WS-SecurityPolicy
Implementation and Deployment Models (2/2)

High Level Architecture

- Signs / encrypt / generates tokens
- Signature verification / decryption / tokens validation

Client

- SecurityHandler
- Request
- Response

Configuration
Deployment descriptor and bindings

AppServer

- SecurityHandler
- Request
- Response

Configuration
Deployment descriptor and bindings

SOAP request+ [WS-Security headers | transport headers]
WS-I Basic Security Profile
Working Group

Deliverables

- Security Scenarios
  - Profile Requirements
  - Message Exchange Patterns (MEPs)

- Basic Security Profile
  - Transport
  - SOAP Messaging
  - Other BP-oriented Considerations
WS-I Basic Security Profile
Working Group

- Profiled Technologies
  - Transport Security
    - HTTP over TLS ("HTTPS") [RFC 2818]
  - SOAP Message Security:
    - WSS: SOAP Message Security v1.0
    - WSS: Username Token Profile
    - WSS: X.509 Certificate Token Profile
    - WSS: Kerberos Token Profile*
  - SOAP Attachment Security
WS-I Basic Security Profile
Working Group

- Coordination with:
  - Basic Profile WG
    - Foundation
    - Attachments
  - Sample Application WG
    - Security Analysis
    - Requirements Definition
    - Design Review
  - Testing Tools WG
    - Tool Analysis
      - Monitor and Analyzer Tools not designed for secure messaging
      - Test Assertion Review
WS-I Security Scenarios

- Requirements
  - Peer Authentication
  - Origin Authentication
  - Integrity
  - Confidentiality

- Mechanisms and Solutions
  - Transport Layer
  - SOAP Message Layer

- Combinations

- Message Exchange Patterns:
  - One-Way
  - Synchronous Request/Response
  - Basic Callback
WS-I Basic Security Profile

- Profile Conformance

  - Artifacts
    - `SECURE_ENVELOPE`
      - SOAP envelope that has been subject to integrity and/or confidentiality protection
    - `SECURE_MESSAGE`
      - Protocol elements that transport a message that have been subject to integrity and/or confidentiality protection

  - Instances
    - `SENDER`
      - Software that generates a message according to the protocol(s) associated with it
    - `RECEIVER`
      - Software that consumes a message according to the protocol(s) associated with it