WSIT Technology

Secure, Reliable, Transactional, and Interoperable Web Services

Sébastien Stormacq
Software Architect
Sun Microsystems, Northern Europe

Agenda

Agenda

- Introduction to Project Metro
- Interoperability Stack
- Security, Trust, SecureConversation
- Security Mechanisms

Basic Terminology

- Authentication: Are you who you say you are?
- Authorization: Are you allowed to have access?
- Integrity: Was message altered before I got it?
- Confidentiality: Only visible to intended recipient ?
- Trust: Do I trust you?
- Auditing: Can I prove what happened?
- Non-repudiation: Can you claim you didn't send the message even if you really did ?

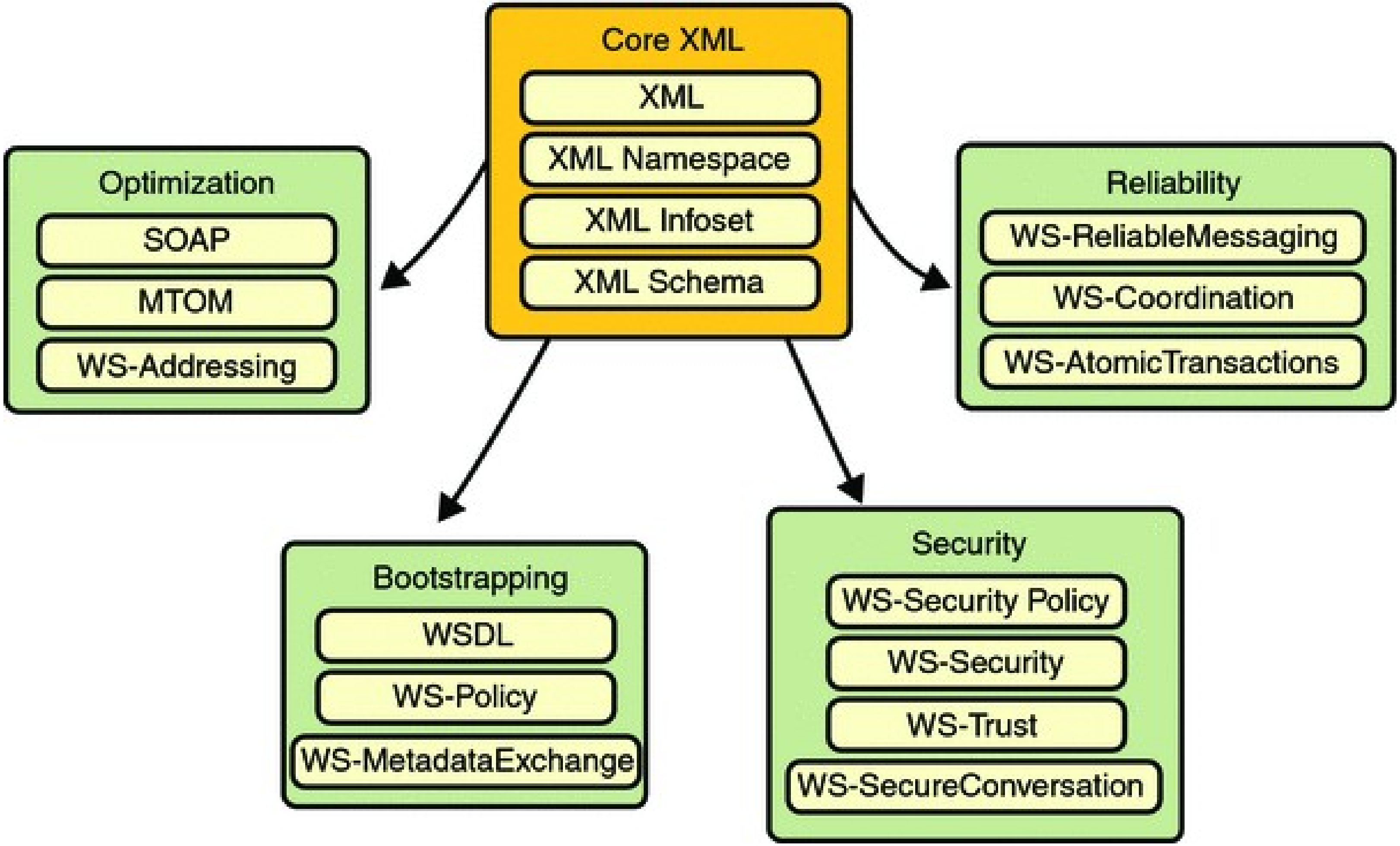
Introduction to Project Metro

- Metro = JAX-WS + WSIT
- Tango → WSIT → Metro
- WSIT (Webservices interoperability technology)
- Enables interoperability with Microsoft .NET 3.0
- Metro – GlassFish Web Services Stack - metro.dev.java.net

Interoperability Stack

- Bootstrapping communication
- End-to-end reliability
- Atomic Transactions
- End-to-end security
- Trust
- Optimized security
- Message Optimization

Interoperability Stack



Core XML

- XML
- XML Namespace
- XML Infoset
- XML Schema

Optimization

- SOAP
- MTOM
- WS-Addressing

Reliability

- WS-ReliableMessaging
- WS-Coordination
- WS-AtomicTransactions

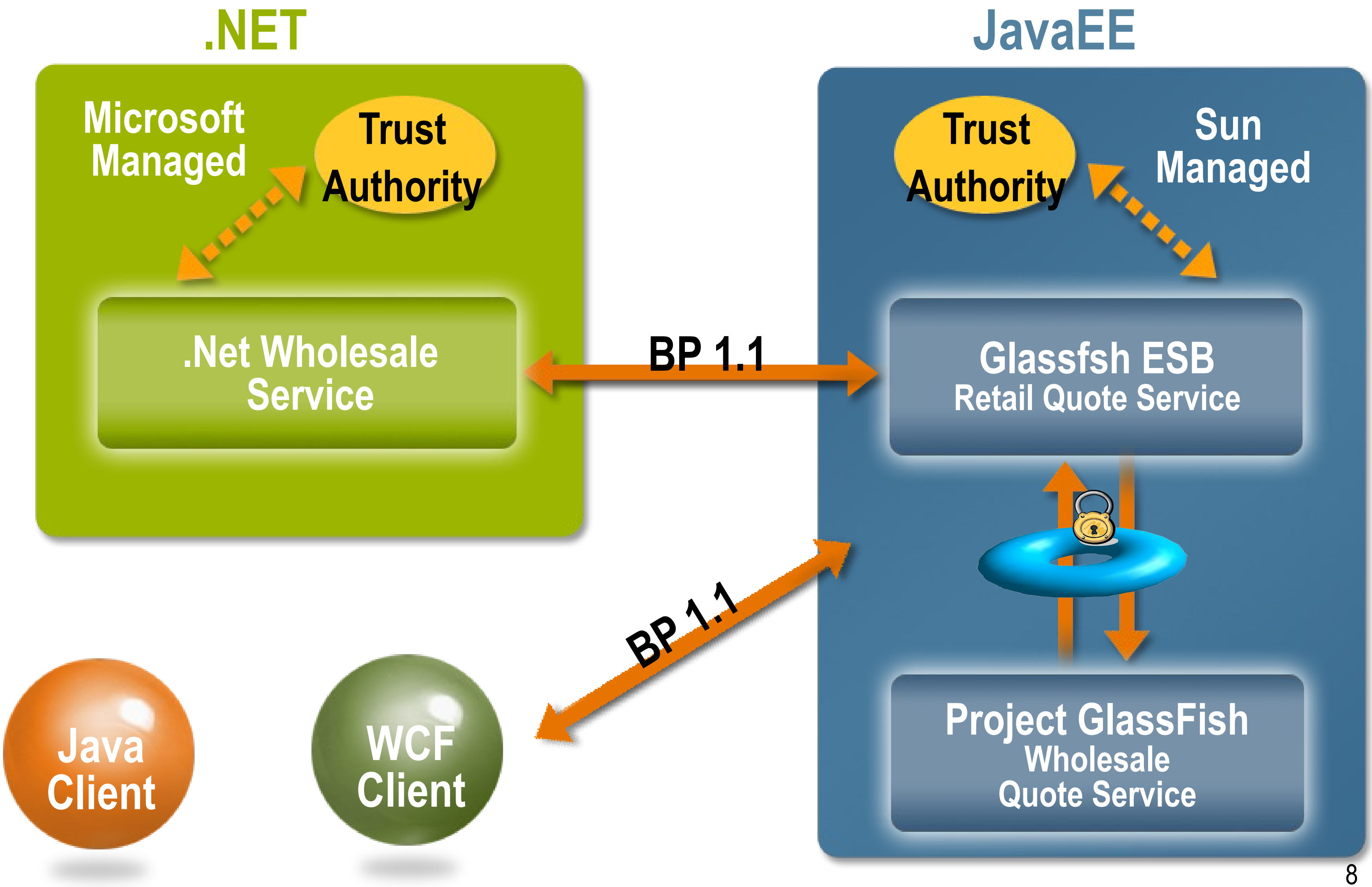
Bootstrapping

- WSDL
- WS-Policy
- WS-MetadataExchange

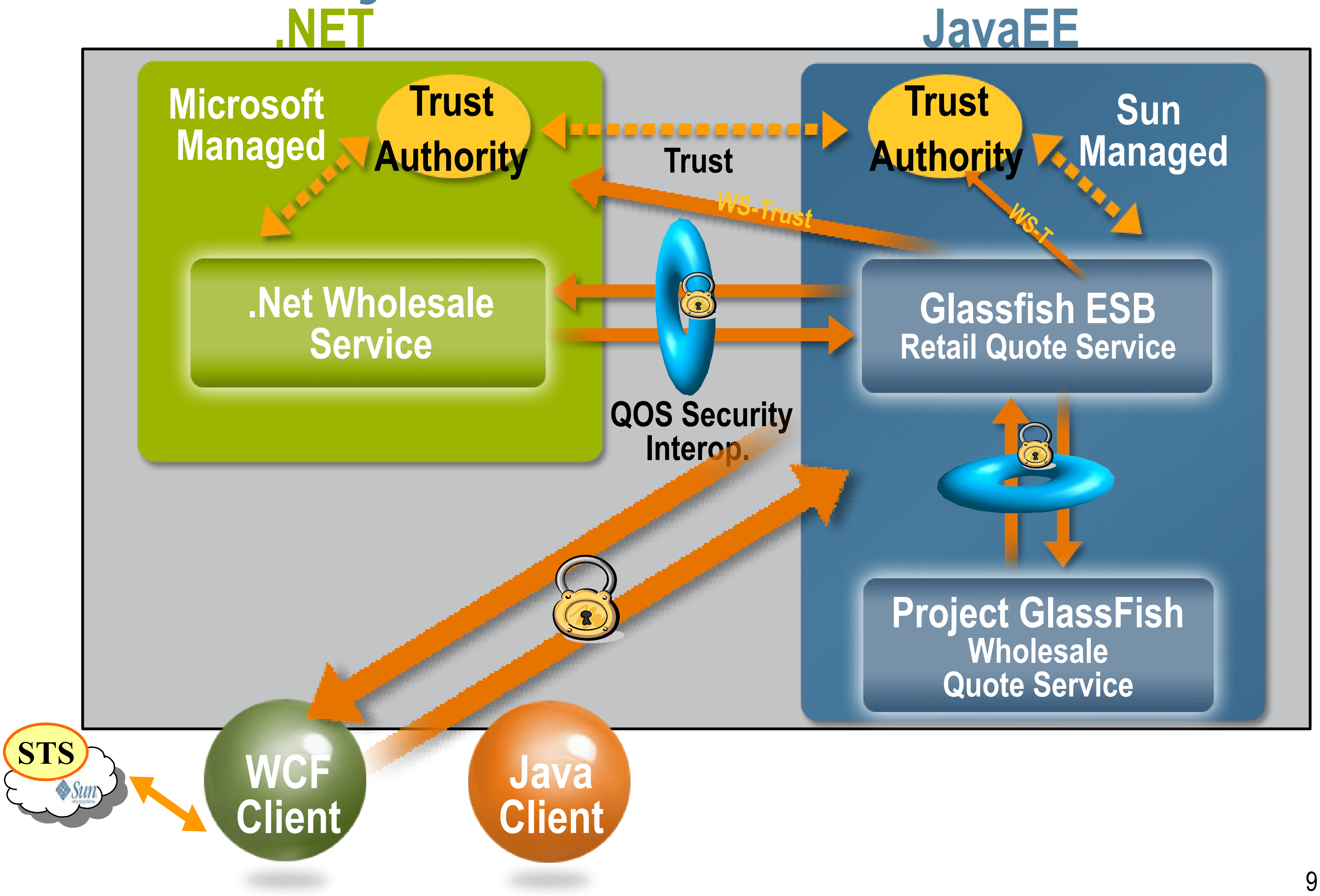
Security

- WS-Security Policy
- WS-Security
- WS-Trust
- WS-SecureConversation

Before Project Metro



With Project Metro



.NET

JavaEE

Microsoft Managed

Trust Authority

Trust

Trust Authority

Sun Managed

WS-Trust

WS-T

.Net Wholesale Service

QOS Security Interop.

Glassfish ESB

Retail Quote Service

Project GlassFish

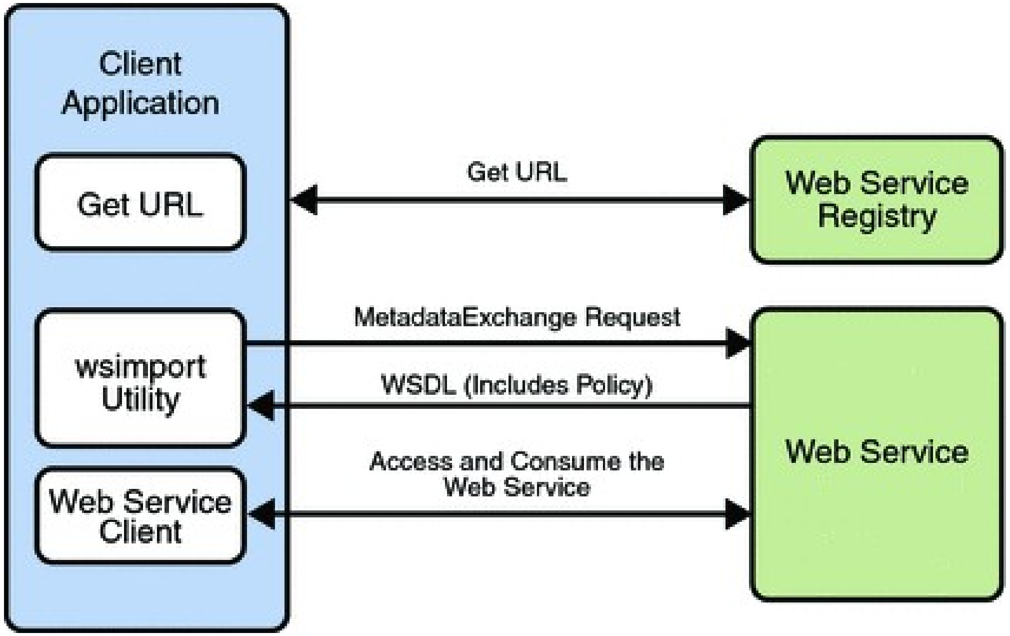
Wholesale Quote Service

STS

WCF Client

Java Client

Bootstrapping Communication



Client Application

Get URL

wsimport Utility

Web Service Client

Get URL

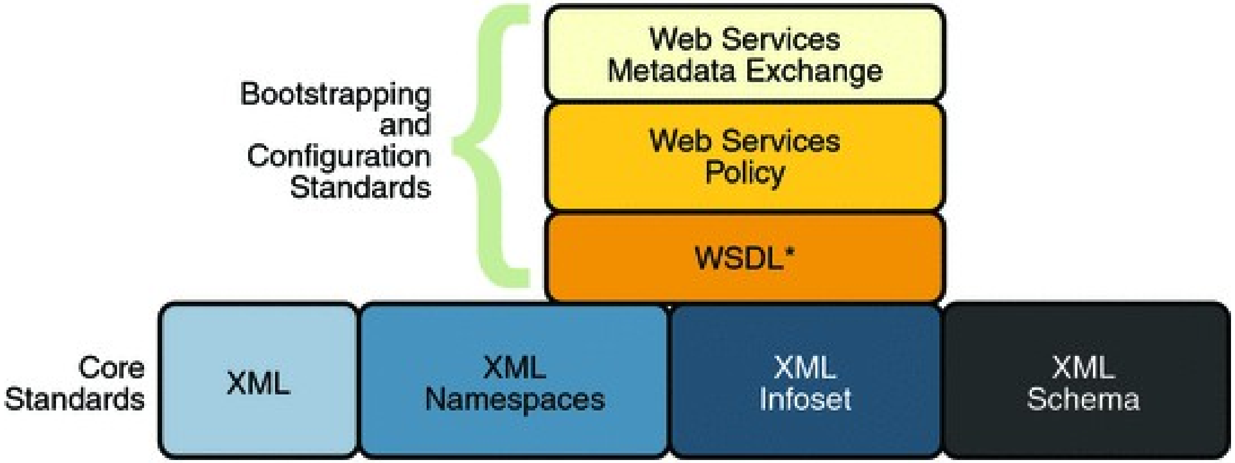
MetadataExchange Request

WSDL (Includes Policy)

Access and Consume the Web Service

Web Service Registry

Web Service



Bootstrapping and Configuration Standards

Web Services Metadata Exchange

Web Services Policy

WSDL*

Core Standards

XML

XML Namespaces

XML Infoset

XML Schema

* Previously implemented in JAX-Web Services

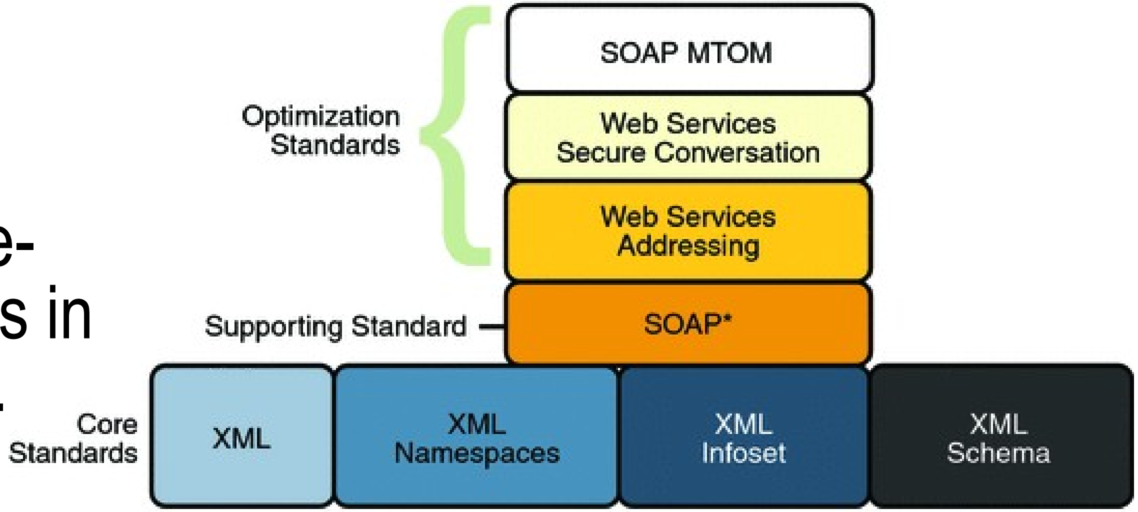
Message Optimisation

- Web Services Addressing
  - > endpoint reference representation.
  - > allows support message transmission in a transport-neutral manner
- Web Services Secure Conversation
  - > efficiency in multiple-message exchanges in a standardized way.

> Message Transmission Optimization Mechanism

> XML-binary Optimized Packaging (XOP),

> selective encoded portions of the SOAP message



Optimization Standards

SOAP MTOM

Web Services Secure Conversation

Web Services Addressing

Supporting Standard

SOAP*

Core Standards

XML

XML Namespaces

XML Infoset

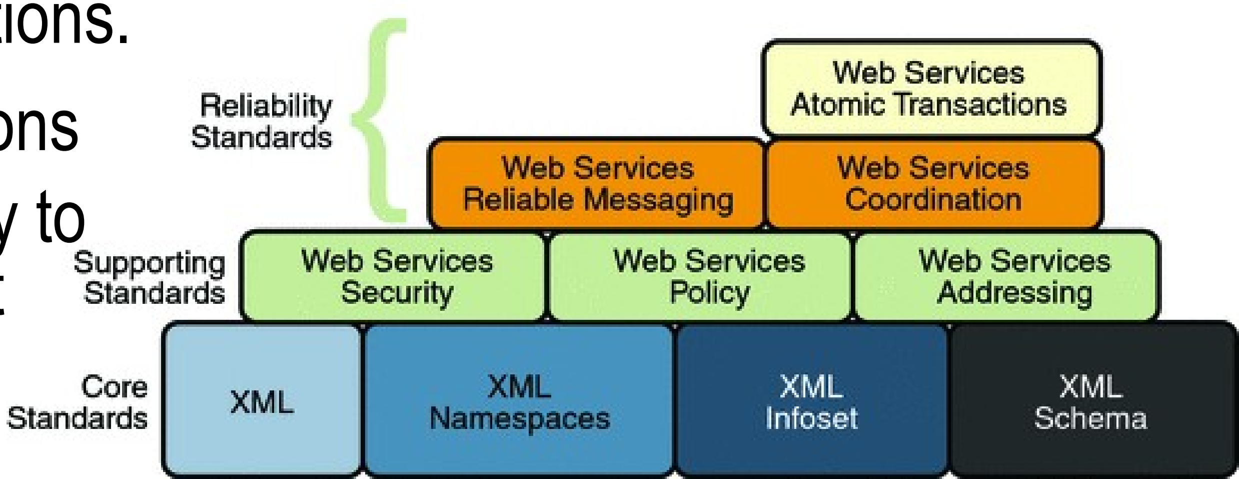
XML Schema

* Previously implemented in JAX-Web Services

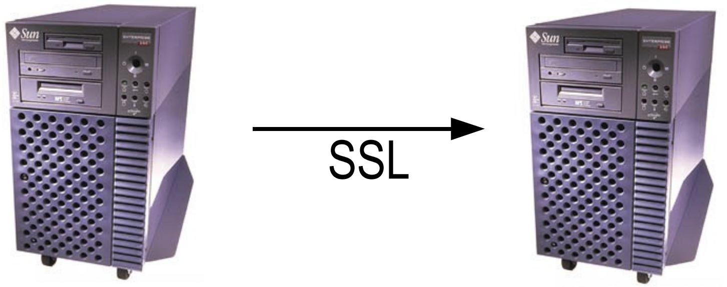
End-to-End Reliability

- WS Reliable Messaging
  - > identify, track, and manage the reliable delivery of messages between exactly two parties
- WS Coordination
  - > a framework for protocols that coordinate the actions of distributed applications.
- WS Atomic Transactions
  - > a standardized way to support XA commit semantics

- Performance cost
  - > Use it when frequent communication failures
  - > loss of application messages in transit
  - > wrong order of application messages



Security



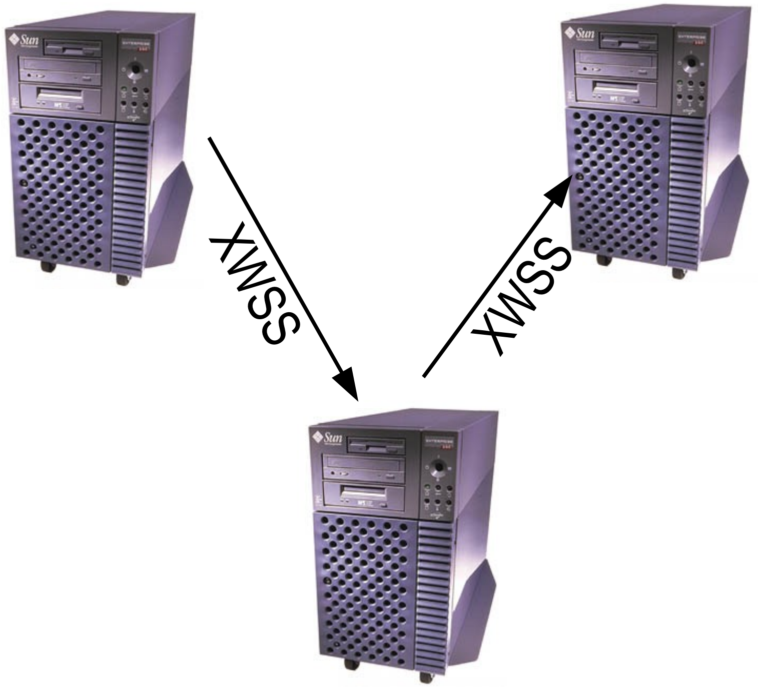
Before WS-Security

SSL/HTTPS

Security at transport layer

All or nothing granularity

Point-to-point



WS-Security

XWSS – XML Web Services Security

Security at SOAP (protocol) layer

Fine granularity possible

Only sign/encrypt credit card #

(e.g., XML subtree)

Works on non-TCP/IP transports

Integrity, Confidentiality, Auth

W3C XML Signature/Encryption

WS Security: SOAP Message Security

- Implements WS-Security 1.1
- Addresses end to end security
- Security Metadata in SOAP header
- Enables **credential exchange**, **message level integrity** and **confidentiality**
- Leverages existing standards and specifications like Security tokens, XML Signature, XML Encryption

WS Security Policy

- Security capabilities and requirements
- Can be attached to the WSDL
- Forces communicating party to secure the message in a certain way
  - Use of certificate and public-private key pair or a valid username/password in order to authenticate
  - Type of key and key sizes and algorithms needed

Security Policy Assertions

- Binding Assertions - Binding level (credentials reqd by client)
- Protection assertions – Operation level (parts and messages)
- Token assertions
- Properties
- Supporting Tokens

Security Policy Binding Assertions

```
<wsp:Policy wsu:Id="Binding_policy">
          ..........
          <sp:SymmetricBinding xmlns:sp="http://...7/securitypolicy">
          <wsp:Policy>
               <sp:ProtectionToken>
                    <wsp:Policy>
                      <sp:X509Token sp:IncludeToken="http://...y/IncludeToken/Never">
                         <wsp:Policy>
                              <sp:RequireDerivedKeys />
                              <sp:RequireThumbprintReference />
                              <sp:WssX509V3Token10 />
                         </wsp:Policy>
                      </sp:X509Token>
                    </wsp:Policy>
               </sp:ProtectionToken>
.....................
</sp:SymmetricBinding>
      <sp:EndorsingSupportingTokens xmlns:
   sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
           <wsp:Policy>
                  <sp:X509Token
   sp:IncludeToken= --------------------------------------------------
```

Server authenticating to client

Req from Client to Server to authenticate

Security Policy Protection Assertions

```
<wsp:Policy wsu:Id="IFinancialService_Input_policy">
    <wsp:ExactlyOne>
        <wsp:All>
            <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
              <sp:Body/>
              <sp:Header Name="To"  namespace="http://www.w3.org/2005/08/addressing"/>
              <sp:Header Name="From"  namespace="http://www.w3.org/2005/08/addressing"/>
              <sp:Header Name="FaultTo"  namespace="http://www.w3.org/2005/08/addressing"/>
              <sp:Header Name="ReplyTo"  namespace="http://www.w3.org/2005/08/addressing"/>
              <sp:Header Name="MessageID"  namespace="http://www.w3.org/2005/08/addressing"/>
              <sp:Header Name="RelatesTo"  namespace="http://www.w3.org/2005/08/addressing"/>
              <sp:Header Name="Action"  namespace="http://www.w3.org/2005/08/addressing"/>
            </sp:SignedParts>
            <sp:EncryptedParts xmlns:
sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
              <sp:Body/>
            </sp:EncryptedParts>
        </wsp:All>
    </wsp:ExactlyOne>
   </wsp:Policy>
```

Security Policy Supporting Token

```
<sp:SignedSupportingTokens xmlns:sp="http://....../securitypolicy">
    <wsp:Policy>
        <sp:UsernameToken
sp:IncludeToken="http://....../IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
                <sp:WssUsernameToken10 />
            </wsp:Policy>
        </sp:UsernameToken>
    </wsp:Policy>
</sp:SignedSupportingTokens>
```

Security Policy – STS Issued SAML Token

```
<sp:IssuedToken sp:IncludeToken=
    "http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/
    IncludeToken/AlwaysToRecipient">
       <sp:Issuer>
           <Address xmlns="http://www.w3.org/2005/08/addressing">
               http://localhost:8080/jaxws-sts/sts
           </Address>
           <Metadata xmlns=
              "http://schemas.xmlsoap.org/ws/2004/09/mex">
               <MetadataSection>
                   <MetadataReference>
                         <Address xmlns=
                         "http://www.w3.org/2005/08/addressing">
                          http://localhost:8080/jaxws-sts/sts
                         </Address>
                    </MetadataReference>
               </MetadataSection>
           </Metadata>
       </sp:Issuer>
</sp:IssuedToken>
```

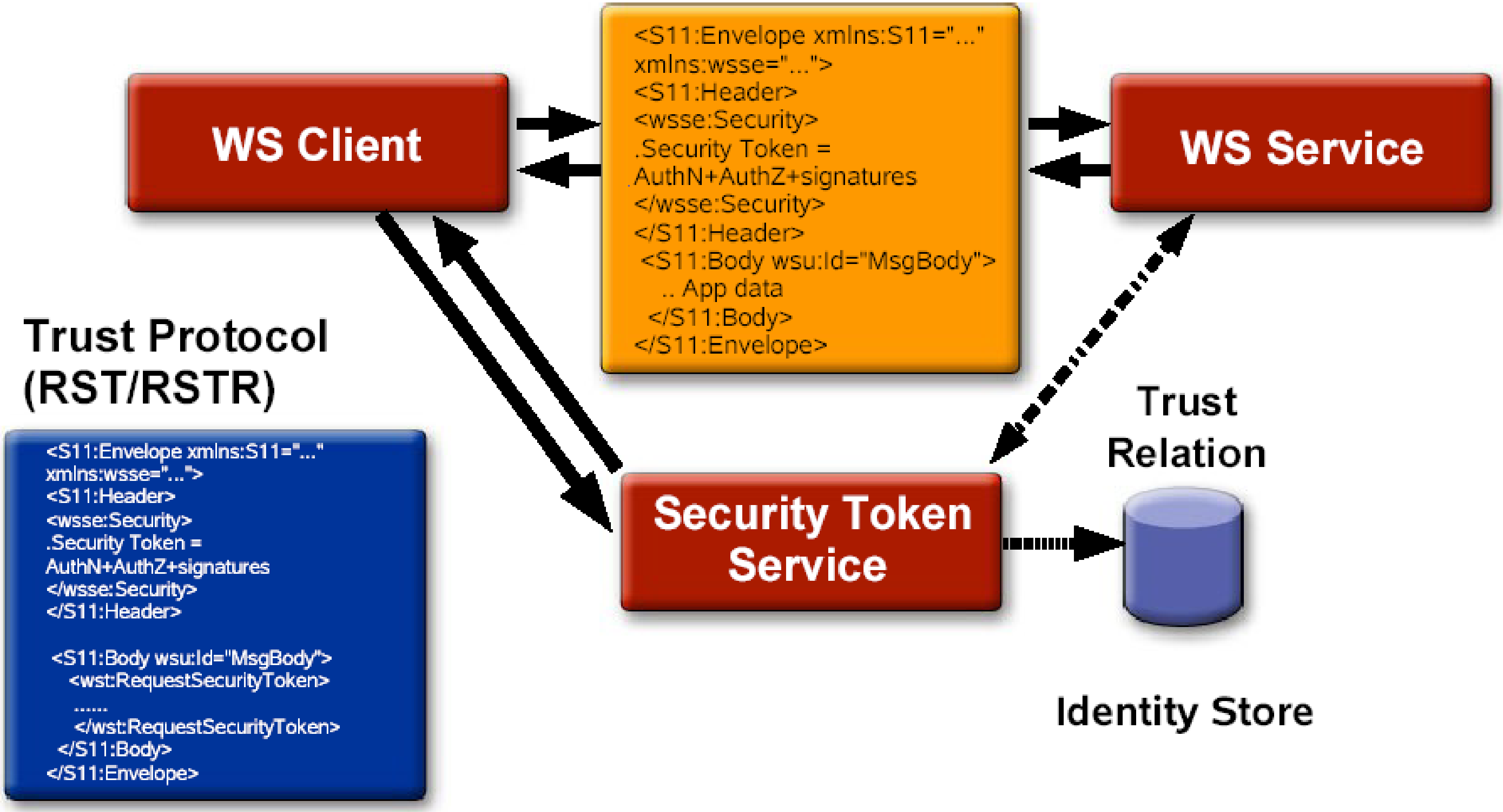
Configurations outside Security Policy

- How to obtain Keys, certificates, username/password from underlying security infrastructure
  - Key Configurations
    - Location of stores, alias of keys
  - Validator/Callback handler Configuration
    - Timestamp, Certificate,Username, Password etc
  - Secure Conversation configuration
    - Session lifetime etc
  - Preconfigured STS
    - Endpoint address, Metadata etc

WS Trust

- Client and service in different security domains
  - No direct trust relationship
  - Required when Server does not trust client or does not understand the client token
- Framework for issuing, renewing, canceling and validating security tokens
- Mechanisms to establish, assess the presence of, and broker trust relationships

WS Trust



```
<S11:Envelope xmlns:S11="..."
xmlns:wsse="...">
<S11:Header>
<wsse:Security>
.Security Token =
AuthN+AuthZ+signatures
</wsse:Security>
</S11:Header>
 <S11:Body wsu:Id="MsgBody">
   .. App data
  </S11:Body>
</S11:Envelope>
```

```
<S11:Envelope xmlns:S11="..."
xmlns:wsse="...">
<S11:Header>
<wsse:Security>
.Security Token =
AuthN+AuthZ+signatures
</wsse:Security>
</S11:Header>

 <S11:Body wsu:Id="MsgBody">
   <wst:RequestSecurityToken>
    ......
    </wst:RequestSecurityToken>
  </S11:Body>
</S11:Envelope>
```

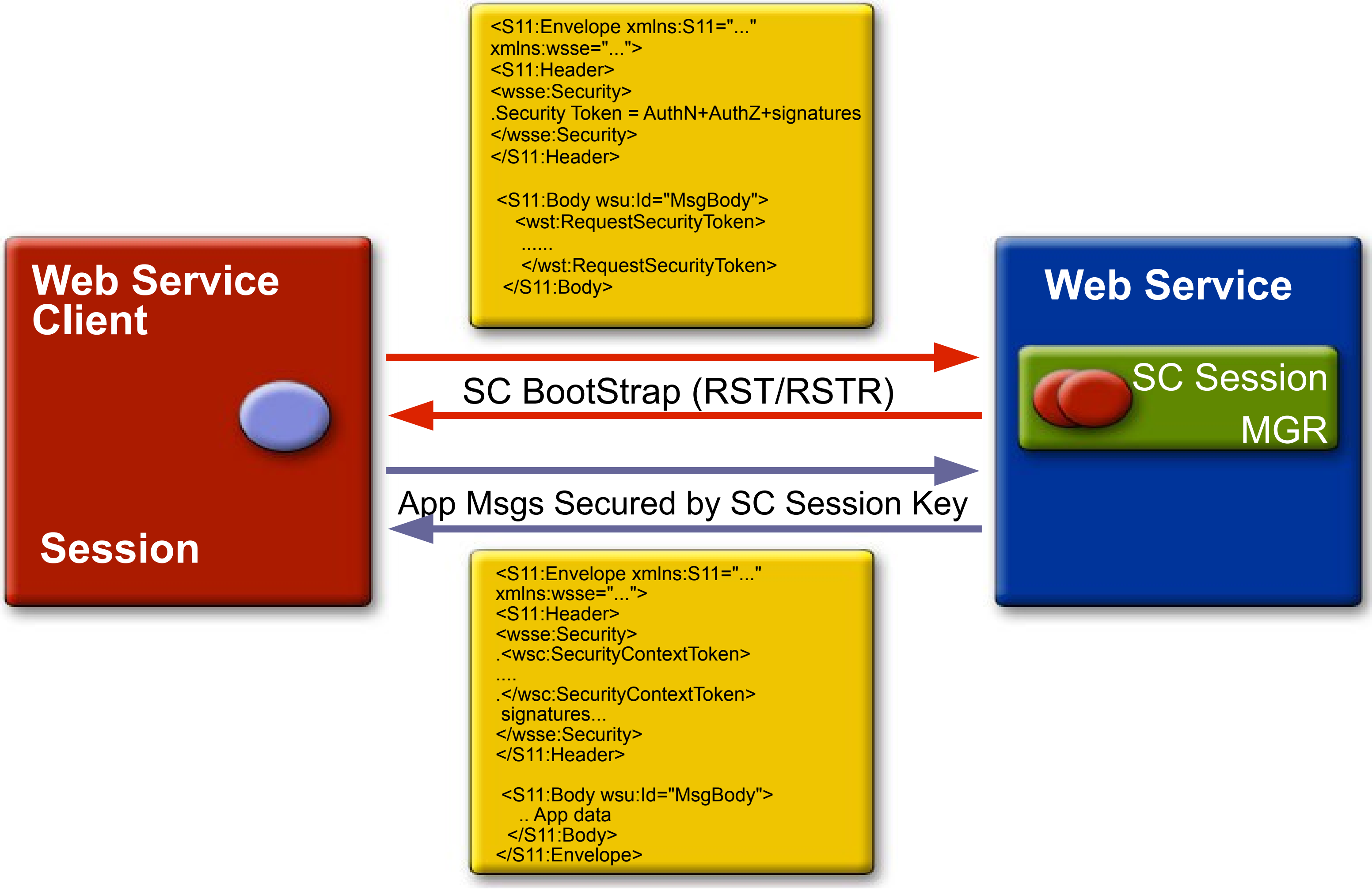
WS Trust Support In Metro

- Client And Service
  - > Authentication and Security with issued tokens from Security Token Services (STS)
  - > Built within the general framework of WS-Security and WS-SecurityPolicy
- Security Token Service
  - > Provide general framework for building STS
  - > Extensible for integrating with existing Access Control, User Identity Management infrastructure for the STS to be used as Identity Provider, Attribute Provider and Authorization Provider.

WS Secure Conversation

- WS Security with multiple messages
  - Performance – authentication, key generation every exchange
  - Security – Frequent credential check is a threat
- Establish Secure session on top of WS Security
- Refers to authentication state and negotiated keys
- Enhances overall security through key derivations
- Improves performance by avoiding repeated key exchange in multi-message exchange scenarios

WS Secure Conversation (Cont.)



```
<S11:Envelope xmlns:S11="..."
xmlns:wsse="...">
<S11:Header>
<wsse:Security>
.Security Token = AuthN+AuthZ+signatures
</wsse:Security>
</S11:Header>

 <S11:Body wsu:Id="MsgBody">
    <wst:RequestSecurityToken>
     ......
     </wst:RequestSecurityToken>
  </S11:Body>
```

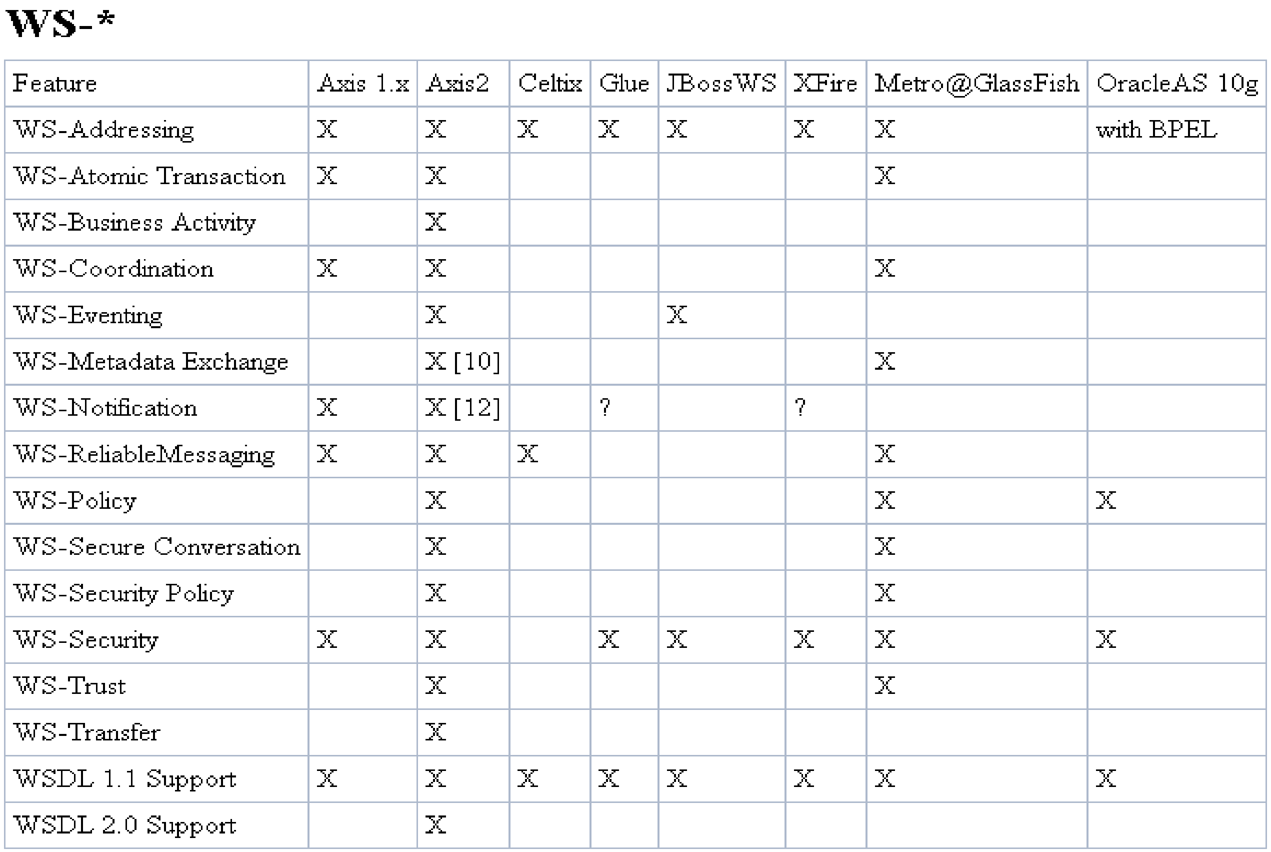
```
<S11:Envelope xmlns:S11="..."
xmlns:wsse="...">
<S11:Header>
<wsse:Security>
.<wsc:SecurityContextToken>
....
.</wsc:SecurityContextToken>
 signatures...
</wsse:Security>
</S11:Header>

 <S11:Body wsu:Id="MsgBody">
    .. App data
  </S11:Body>
</S11:Envelope>
```

Security Mechanisms

- Username Authentication with Symmetric Keys
- Mutual Certificates Security
- Transport Security (SSL)
- Message Authentication over SSL
- SAML Authorization over SSL
- Endorsing Certificate
- SAML Sender Vouches with Certificates
- SAML Holder of Key
- STS Issued Token
- STS Issued Token with Service Certificate
- STS Issued Endorsing Token

Comparison to Other WS Stacks



WS-*

Feature	Axis 1.x	Axis2	Celtix	Glue	JBossWS	XFire	Metro@GlassFish	OracleAS 10g
WS-Addressing	X	X	X	X	X	X	X	with BPEL
WS-Atomic Transaction	X	X					X	
WS-Business Activity		X						
WS-Coordination	X	X					X	
WS-Eventing		X			X			
WS-Metadata Exchange		X [10]					X	
WS-Notification	X	X [12]		?		?		
WS-ReliableMessaging	X	X	X				X	
WS-Policy		X					X	X
WS-Secure Conversation		X					X	
WS-Security Policy		X					X	
WS-Security	X	X		X	X	X	X	X
WS-Trust		X					X	
WS-Transfer		X						
WSDL 1.1 Support	X	X	X	X	X	X	X	X
WSDL 2.0 Support		X						

http://wiki.apache.org/ws/StackComparison?action=print

Metro and Http BC

- When to use Metro in Glassfish ESB
  - Java EE based WS (EJB, Web App WS)
  - No need to composing services
- Value add from Http BC
  - Composing services, orchestrating using BPEL, need to access WS that required WS-*
  - Making your BP interoperable (securing, transaction etc)
  - Outside BPEL
    - XSLT SE, Database BC,
    - BC to BC mediation

Thank You

Sébastien Stormacq
sebastien.stormacq@sun.com
http://blogs.sun.com/sebsto

Server Side Configuration Requirements

Mechanism	Keystore	Truststore	STS	SSL	User in Glassfish
Username Authentication with Symmetric Keys	YES				YES
Mutual Certificates Security	YES	YES			
Transport Security (SSL)				YES	YES
Message Authentication over SSL – Username Token				YES	YES
Message Authentication over SSL – X.509 Token		YES		YES	
SAML Authorization over SSL	YES	YES		YES	
Endorsing Certificate	YES	YES			
SAML Sender Vouches with Certificates	YES	YES			
SAML Holder of Key	YES	YES			
STS Issued Token	YES	YES	YES		
STS Issued Token with Service Certificate	YES	YES	YES		
STS Issued Endorsing Token	YES	YES	YES		

Client Side Configuration Requirements

Mechanism	Keystore	Truststore	Default User	SAML Callback Handler	STS	SSL	User in Glassfish
Username Authentication with Symmetric Keys		YES	YES				YES
Mutual Certificates Security	YES	YES					
Transport Security (SSL)						YES	YES
Message Authentication over SSL – Username Token			YES			YES	YES
Message Authentication over SSL – X.509 Token	YES					YES	
SAML Authorization over SSL	YES	YES		YES		YES	
Endorsing Certificate	YES	YES					
SAML Sender Vouches with Certificates	YES	YES		YES			
SAML Holder of Key	YES	YES		YES			
STS Issued Token	YES	YES			YES		
STS Issued Token with Service Certificate	YES	YES			YES		
STS Issued Endorsing Token	YES	YES			YES		

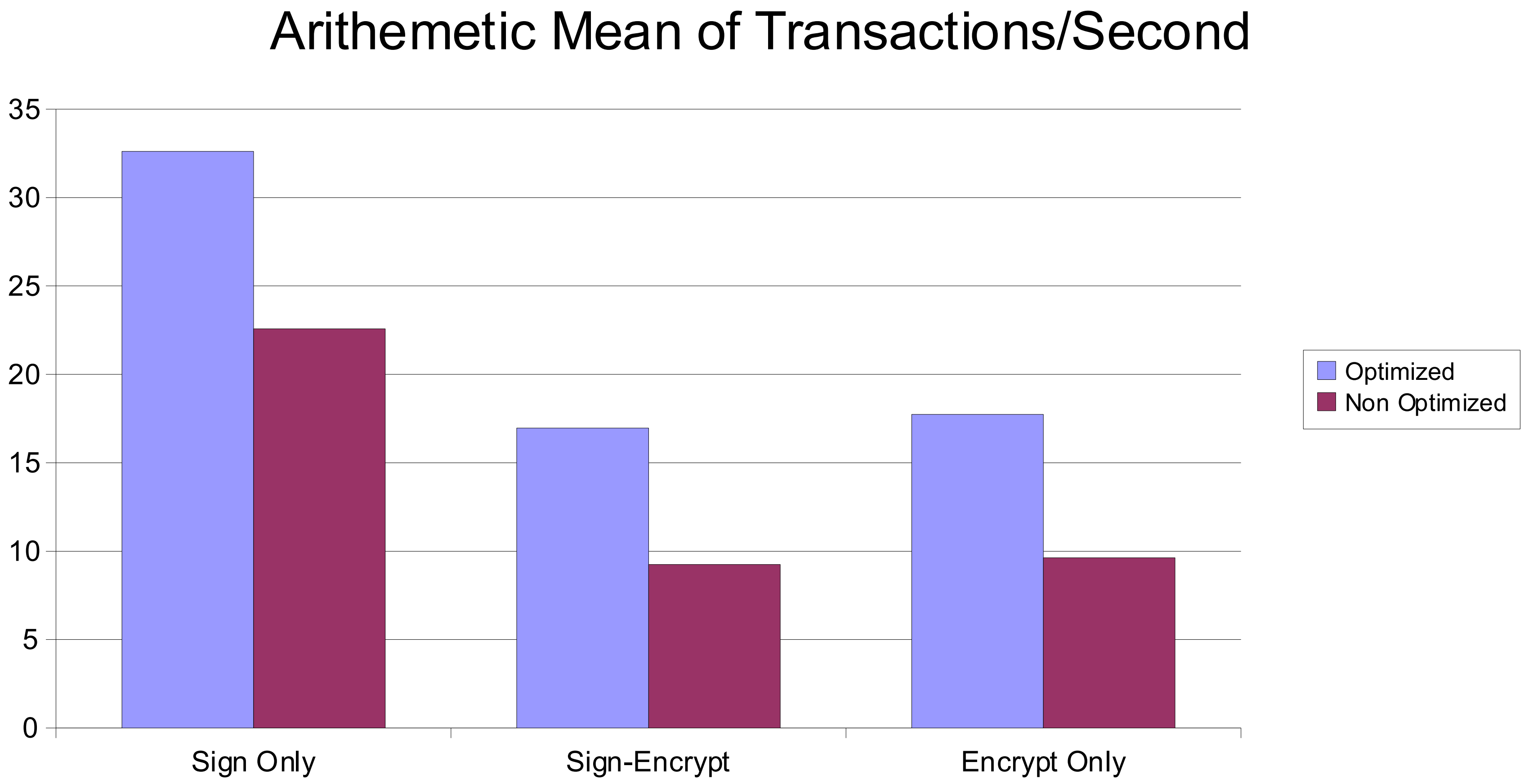
WS Security

- Security Specification Supported
  - Web Services Security (versions 1.0 and 1.1)
  - WS Security Policy
  - WS Secure Conversation
  - WS Trust
- Interoperability
- Integrated into GlassFish / AppServer through JSR 196
- Performance
- NetBeans and Tooling

Optimizing the performance with security

- Security - CPU intensive and increase message size
- Traditionally - convert SOAP to DOM to perform Signature/Encryption
  - Large amount of objects are allocated
  - Memory consumption
- New implementation for Security
  - Security operations in Streaming fashion
  - Switch back to DOM for Xpath
  - Exciting results – 40 to 200 % improvement

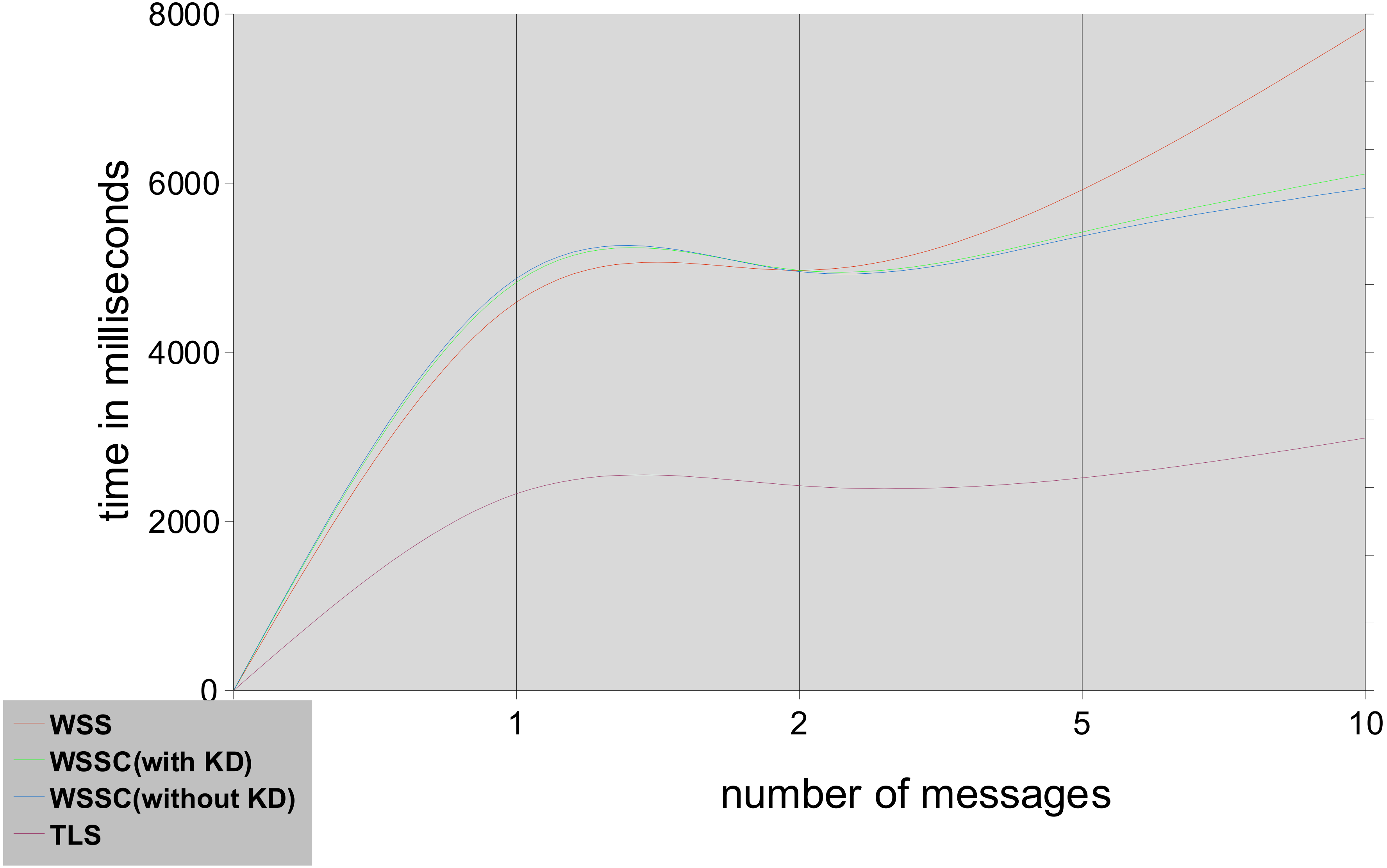
Optimizing the performance with security



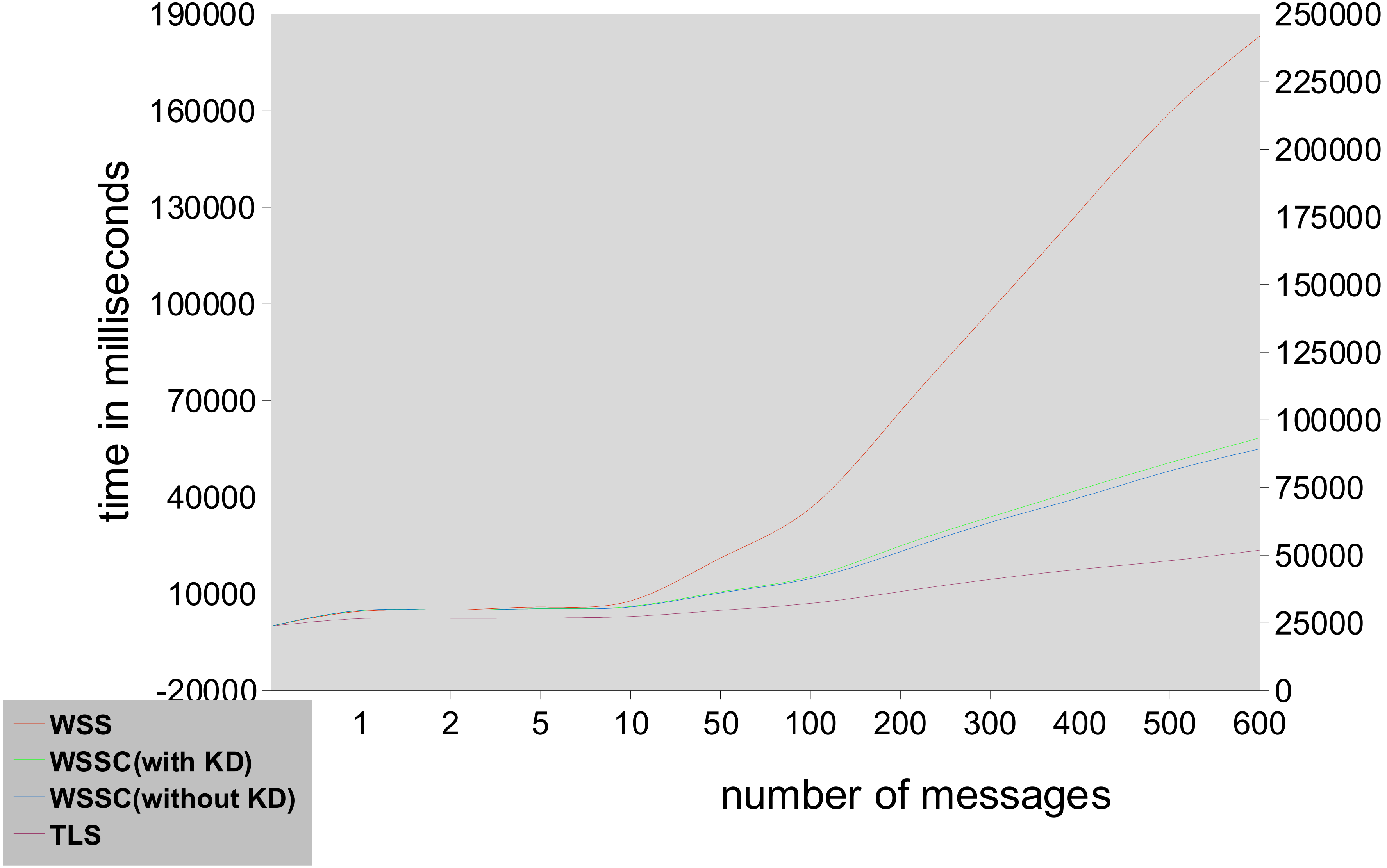
Optimizing the performance with security

- WSS 1.1 allows Encrypted Keys to be reused by the server
- Secure Conversation for multiple message exchanges (Will discuss in coming slides)
  - No need to reauthenticate or do public key crypto repeatedly
  - Additional 250 to 350% improvement

WSS vs WSSC vs TLS



WSS vs WSSC vs TLS



Security Token Service

- BaseSTS is used to implement the actual STS as a Web Service
- Authentication and secure communication between client and STS handled in the same way as for a regular Web Service.
- Support for issuing SAML 1.0, SAML 1.1, SAML 2.0 assertions by default.
- Support for issuing public and symmetric proof keys
- Extensible to support for issuing other types of tokens.

Security Token Service : Extension Points

- Allows for plugging-in authorization mechanisms for controlling the issuing of the tokens according to the user's identity for the targeted services. You can't blindly issue a token to anybody for any service.
- Allows for plugging-in user mappings for controlling the user identity/attributes carried in the SAML token issued by STS for different services. These identity/attributes can be used by the targeted services for authentication/authorization purpose.