

# Web Services Security: What's Required To Secure A Service-Oriented Architecture

*An Oracle White Paper  
January 2008*

# Web Services Security: What's Required To Secure A Service-Oriented Architecture.

## INTRODUCTION

The service-oriented architecture (SOA) concept is now embraced by many companies worldwide. However, because of its nature (loosely-coupled connections) and its use of open access (HTTP), SOA adds a new set of requirements to the security landscape.

Many companies rely on the Secure Socket Layer (SSL) protocol to protect access to SOA deployments. SSL provides authentication, confidentiality and message integrity. However, when the data is not "in transit," the data is not protected, which makes the environment vulnerable to attacks in multi-step transactions. As a result, there is a need to address more specific SOA security challenges by relying on additional, application-level security.

Application-level security is mainly defined by industry standards. Some of these standards have been around for several years, originally designed for web applications and later leveraged by SOA, for example SSL (mentioned above), and Kerberos, a cross-platform authentication and single sign-on system.

Other standards have specifically been created to provide security to networks of web services, for example WS-Security and WS-Policy.

The purpose of this paper is to describe the standards that are key to providing secure SOA deployments using web services.

## EVOLVING REQUIREMENTS

Until recently, the onus was on the developer to tackle web services security. Developers used to code security into web services thus creating environment "silos" difficult to manage and costly to maintain.

SOA deployments have become more and more complex, creating additional challenges that developers alone cannot meet anymore, such as cryptographic data protection, identity management, and governance.

Vendors such as Oracle have created solutions to help enterprises meet these new challenges. However, companies do have legacy environments and existing infrastructures that must accommodate the SOA paradigm, and a single vendor may not be able to provide every single piece of the SOA security, management,

Because they are predicated on industry standards, Oracle Fusion Middleware components can be "hot-pluggable" with other standards-compliant vendor platforms.

and governance puzzle. As a result, vendors must rely on industry standards to make their offerings interoperable.

Oracle implements key industry standards in its products to make them “hot pluggable” and in many cases Oracle is a driving force behind new standards designed to meet ever-growing SOA security and management challenges.

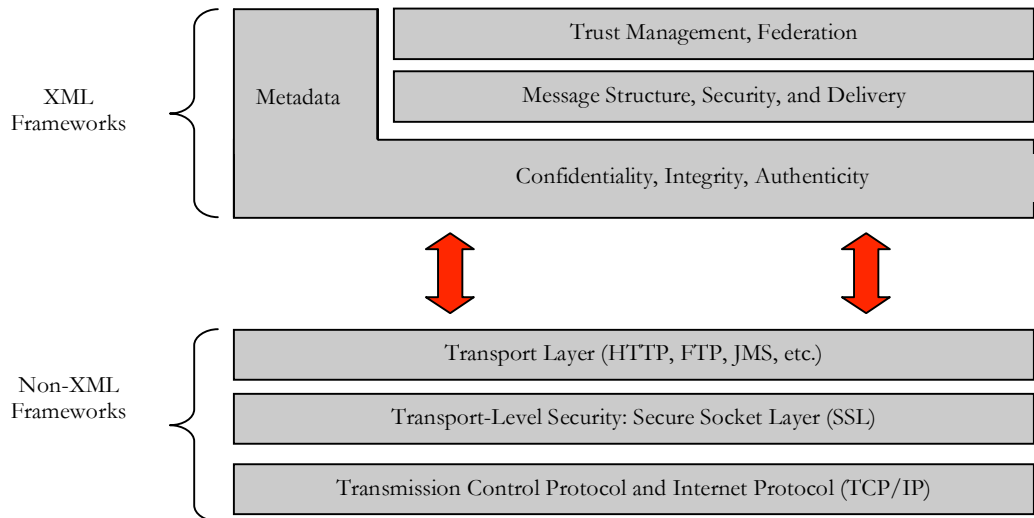
### SOA SECURITY STANDARDS

The main goal of SOA security standards is to provide a basis for interoperability among multiple products used in heterogeneous customer environments.

Standards-based implementation strategies accelerate development, simplify integration, and reduce administrative costs over time.

### Security Standards Topology

Security standards are implemented in non-XML frameworks at the transport level, and in XML frameworks at the application level, as shown in the figure below.



- Confidentiality, Integrity, Authenticity: XML Encryption, XML Signature.
- Message-Level Security: WS-Security.
- Secure Message Delivery: WS-Addressing, WS-ReliableMessaging.
- Metadata: WS-Policy, WS-SecurityPolicy.
- Trust Management: SAML, WS-Trust, WS-SecureConversation, WS-Federation.
- Public Key Infrastructure: PKCS, PKIX, XKMS

Most SOA industry standards are defined in XML frameworks. The last few years have seen the emergence of a plethora of XML-based specifications addressing various aspects of SOA security. Most of these specifications are part of the so-called WS-\* (Web Services specifications) stack.

The following sections describe the standards that are key to providing secure and manageable SOA environments.

### **Transport-Level Security: SSL**

Secure Socket Layer (SSL), otherwise known as Transport Layer Security (TLS), the Internet Engineering Task Force (IETF) officially standardized version of SSL, is the most widely used transport-level data-communication protocol providing:

- Authentication -- the communication is established between two trusted parties,
- Confidentiality -- the data exchanged is encrypted,
- Message Integrity -- the data is checked for possible corruption,
- Secure key exchange between client and server.

SSL can be used in three modes:

- No authentication: Neither the client nor the server authenticates itself to the other. No certificates are sent or exchanged. In this case, only confidentiality (encryption / decryption) is used.
- One-way authentication (or server authentication): Only the server authenticates itself to the client. The server sends the client a certificate verifying that the server is authentic. This is typically the approach used for Internet transactions such as online banking.
- Two-way authentication (or bilateral authentication): Both client and server authenticate themselves to each other by sending certificates to each other. This approach is necessary to prevent attacks from occurring between a proxy and a web service endpoint. For example, two-way authentication could be used between Oracle Web Services Manager's Gateway and a remote web service.

SSL uses a combination of secret-key and public-key cryptography to secure communications. SSL traffic uses secret keys for encryption and decryption, and the exchange of public keys is used for mutual authentication of the parties involved in the communication.

### **Application-Level Security: XML Frameworks**

Most WS-\* specifications started as proprietary industry initiatives. Some of these specifications (e.g., WS-Security, WS-Trust, WS-Policy) have been transferred over to standards bodies such as the Organization for the Advancement of Structured Information Standards (OASIS) or the World Wide Web Consortium (W3C).

Transport-level and application-level security approaches are complementary.

They can be used separately or (more efficiently) together.

WS-\* specifications often depend on each other, for example, WS-Policy is used in conjunction with WS-Security. WS-\* specifications also leverage non-WS-\* specifications, for example, WS-Security uses XML Encryption and XML Signature.

### **XML Encryption (Confidentiality)**

XML Encryption defines:

- How digital content is encrypted and decrypted,
- How the encryption key information is passed to a recipient,
- How encrypted data is identified to facilitate decryption.

An XML document may be encrypted as a whole or in part as shown in the following example, where only the credit card number is encrypted.

```
<?xml version="1.0"?>
<PaymentInfo xmlns="http://www.example.com/payment">
  <CreditCard>
    <Name>Marc</Name>
    <CreditCardNumber>
      <EncryptedData xmlns="http://www..." Type="http://www...">
        <CipherData>
          <CipherValue>A23B4...5C56</CipherValue>
        </CipherData>
      </EncryptedData>
    </CreditCardNumber>
    <ExpireDate>03/2011</ExpireDate>
  </CreditCard>
</PaymentInfo>
```

### **XML Signature (Integrity, Authenticity)**

XML Signature binds the sender's identity (or "signing entity") to an XML document. The document is signed using the sender's private key, the signature is verified using the sender's public key.

Signing and signature verification can be done using asymmetric or symmetric keys. XML Signature also ensures non-repudiation of the signing entity, i.e., it provides a proof that messages have not been altered since they were signed.

A signature can apply to a whole document or just part of a document, as shown in the following example.

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <!-- The signedInfo element allows us to sign any portion of a
  document -->
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www..."/>
    <SignatureMethod Algorithm="http://www..."/>
    <Reference URI="#Body">
      <DigestMethod Algorithm="http://www..."/>
      <DigestValue>o+jtqlieRtF6DrUb...X809M/CmySg</DigestValue>
    </Reference>
  </SignedInfo>
  <!-- Following is the result of running the algorithm over the
  document. If changes are made to the document, the SignatureValue is
  changed. The security application verifies the SignatureValue,
  extracts the X.509 cert and uses it to authenticate the user -->
  <SignatureValue>oa+ttbsvSFi...EtRD2oNC5</SignatureValue>
```

```

<KeyInfo>
  <KeyValue>
    <!-- Following is the public key that matches the private key
    that signs the document -->
    <RSAKeyValue>
      <Modulus>5TT/oolzTiP++Ls6GLQUM8xoFFrAlZQ...</Modulus>
      <Exponent>EQ==</Exponent>
    </RSAKeyValue>
  </KeyValue>
  <!-- Following is the certificate -->
  <X509Data>
    <X509Certificate>wDCCAXqgAwIBAgI...</X509Certificate>
  </X509Data>
</KeyInfo>
</Signature>

```

### Web Services Security (WS-Security)

WS-Security is arguably the most important WS-\* specification. WS-Security is used with virtually all of the other WS-\* specifications.

WS-Security specifies SOAP security extensions that provide confidentiality using XML Encryption and data integrity using XML Signature.

WS-Security also includes profiles that specify how to insert different types of binary and XML security tokens in WS-Security headers for authentication and authorization purposes:

WS-Security is the most important specification for securing SOA.

It is supported by all market-leading vendors, thus facilitating cross-product interoperability.

- Username with Password (defines how a web service consumer can supply a username as a credential for authentication).
- X.509 Certificate (a signed data structure designed to send a public key to a receiving party).
- Kerberos ticket (a binary authentication and session token).
- Security Assertion Markup Language (SAML) assertion (see more detail on SAML later in this document).
- REL document (rights expression language (REL) license tokens inserted in WS-Security headers are used for authorization).

The most prevalent security tokens used with WS-Security are Username, X.509 Certificates, SAML assertions, and Kerberos tickets (all supported by Oracle Web Service Manager).

The following paragraphs provide a short description of X.509 certificates, SAML, and Kerberos.

#### X.509 Certificates

An X.509 digital certificate is a signed data structure designed to send a public key to a receiving party. A certificate includes standard fields such as certificate ID, issuer's Distinguished Name (DN), validity period, owner's DN, owner's public key, etc.

Certificates are issued by certificate authorities (CA), for example Verisign or Thwate. A CA verifies an entity's identity and grants a certificate, signing it with the CA's private key. The CA publishes its own certificate which includes its public key.

Each network entity has a list of the certificates of the CAs it trusts. Before communicating with another entity, a given entity uses this list to verify that the signature of the other entity's certificate is from a trusted CA.

### **SAML**

SAML is widely adopted by the industry, both for browser-based federation and federation enabled by web services flows.

The Security Assertion Markup Language (SAML) is an open framework for sharing security information on the Internet through XML documents. SAML was designed to address the following:

- Limitations of web browser cookies to a single domain: SAML provides a standard way to transfer cookies across multiple Internet domains.
- Proprietary web single sign-on (SSO): SAML provides a standard way to implement SSO within a single domain or across multiple domains. This functionality is provided by the Oracle Identity Federation product.
- Federation: SAML facilitates identity management (e.g., account linking when a single user is known to multiple web sites under different identities), also supported by Oracle Identity Federation.
- Web Services Security: SAML provides a standard security token (a SAML assertion) that can be used with standard web services security frameworks (e.g., WS-Security) – This is the use of SAML that is particularly relevant to web services security, fully supported by Oracle Web Services Manager.
- Identity propagation: SAML provides a standard way to represent a security token that can be passed across the multiple steps of a business process or transaction, from browser to portal to networks of web services, also a feature supported by Oracle Web Services Manager.

The SAML framework includes 4 parts:

- Assertions: How you define authentication and authorization information.
- Protocols: How you ask (SAML Request) and get (SAML Response) the assertions you need.
- Bindings: How SAML Protocols ride on industry-standard transport (e.g., HTTP) and messaging frameworks (e.g., SOAP).
- Profiles: How SAML Protocols and Bindings combine to support specific use cases.

In the context of WS-Security, only SAML assertions are used. The protocols and bindings are provided by the WS-Security framework.

The reason why SAML assertions are very popular security tokens within WS-Security is because they are very expressive and can help prevent man-in-the-middle and replay attacks.

Typically, a SAML assertion makes statements about a principal (a user or an application). All SAML assertions include the following common information:

- Issuer ID and issuance timestamp
- Assertion ID
- Subject
- Name
- Optional subject confirmation (e.g., a public key)
- Optional conditions (under which an assertion is valid)
- Optional advice (on how an assertion was made)

SAML assertions can include three types of statements:

- *Authentication statement*: issued by an authentication authority upon successful authentication of a subject. It asserts that Subject S was authenticated by Means M at Time T.
- *Attribute statement*: issued by an attribute authority, based on policies. It asserts that Subject S is associated with Attributes A, B, etc. with values a, b, etc.
- *Authorization decision statement* (deprecated in SAML 2.0, now supported by XACML): issued by an authorization authority which decides whether to grant the request by Subject S, for Action A (e.g., read, write, etc.), to Resource R (e.g., a file, an application, a web service), given Evidence E.

SAML assertions can be embedded (i.e., a SAML assertion can contain another SAML assertion). SAML assertions can be signed (using XML Signature) and/or encrypted (using XML Encryption).

### **Kerberos**

Kerberos is a cross-platform authentication and single sign-on system. The Kerberos protocol provides mutual authentication between two entities relying on a shared secret (symmetric keys).

Kerberos uses particular terminology:

- A *Principal* is an identity for a user (i.e., a user is assigned a principal), or an identity for an application offering Kerberos services.
- A *Realm* is a Kerberos server environment; a Kerberos realm can be a domain name such as EXAMPLE.COM (by convention expressed in uppercase).

Kerberos involves a client, a server, and a trusted party to mediate between them called the Key Distribution Center (KDC). Each Kerberos realm has at least one

KDC. KDCs come in different packages based on the operating platform used (for example, on Microsoft Windows, the KDC is a domain service).

The Kerberos Token profile of WS-Security allows business partners to use Kerberos tokens in service-oriented architectures.

### **Inserting Security Tokens in WS-Security headers**

Username, X.509, SAML, and Kerberos security tokens are inserted in WS-Security headers in a standard way (defined in the WS-Security token profiles specifications).

For example, a SOAP message using a SAML assertion would look like this:

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>
    <wsse:Security>
      <saml:Assertion>...</saml:Assertion>
    </wsse:Security>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>...</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In this case, SAML assertions and references to assertion identifiers are contained in the `<wsse:Security>` element, which in turn is included in the `<SOAP-ENV:Header>` element (described in the WS-Security SAML Token Profile).

SAML security tokens can be used in various situations (or scenarios), based on specific requirements. For example, Oracle Web Services Manager supports the SAML Holder-of-Key or SAML Sender-Vouches scenarios described in the WS-Security SAML Token Profile.

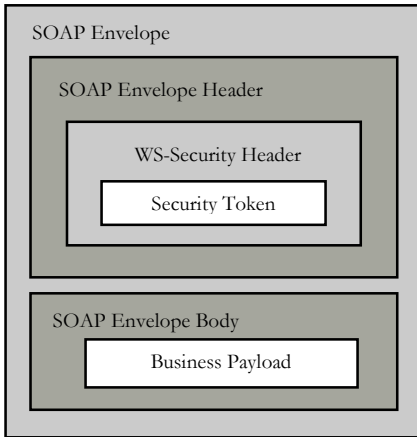
### **Web Services Addressing (WS-Addressing)**

SOAP does not provide a standard way to specify where a message is going or how responses or faults are returned. WS-Addressing provides an XML framework for identifying web services endpoints and for securing end-to-end endpoint identification in messages.

A web service endpoint is a resource (such as an application or a processor) to which web services messages are sent. Oracle Web Services Manager, for example, uses WS-Addressing for metric correlation.

Following is an example using WS-Addressing (`wsa` is the namespace for WS-Addressing):

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <S:Header>
    <wsa:MessageID>http://example.com/xyz-abcd-123</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://example.myClient1</wsa:Address>
    </wsa:ReplyTo>
  </S:Header>
  etc.
```



WS-Addressing is transport-independent, i.e., the request may be over JMS and the response over HTTP. WS-Addressing is used with other WS-\* specifications, such as WS-Policy.

#### **Web Services Reliable Messaging Protocol (WS-ReliableMessaging or WS-RM)**

WS-RM is often confused with WS-Reliability. WS-RM and WS-Reliability have a very similar purpose (i.e., the guarantee that a message that has been sent will actually be delivered) and they may be merged in the future, however, today WS-RM seems to hold sway.

*(Note: Oracle Application Server 10g supports WS-Reliability whereas Oracle Application Server 11g will support WS-RM.)*

WS-RM defines a framework for identifying and managing the reliable delivery of messages between web services endpoints. WS-RM is predicated on the SOAP messaging structure (SOAP binding) and relies on WS-Security, WS-Policy, and WS-Addressing to provide reliable messaging.

WS-RM defines a reliable messaging (RM) source (the party that sends the message) and an RM destination (the party that receives the message). WS-RM mandates prerequisites, for example, trust between endpoints must be established, and the message and endpoints must be formally identified (this is achieved through the use of the complementary WS-\* specifications mentioned earlier).

WS-RM Policy defines a policy assertion that leverages the WS-Policy framework in order to enable an RM destination and an RM source to describe their requirements for a given sequence.

WS-Policy is emerging as one of the key metadata standards in web services security.

#### **Web Services Policy (WS-Policy)**

Together with WS-Security, WS-Policy is another key industry standard for Oracle Fusion Middleware security.

A web service provider may define conditions (or policies) under which a service is to be provided. The WS-Policy framework enables one to specify policy information that can be processed by web services applications, such as Oracle Web Services Manager.

A policy is expressed as one or more policy assertions representing a web service's capabilities or requirements. For example, a policy assertion may stipulate that a request to a web service be encrypted. Likewise, a policy assertion can define the maximum message size that a web service can accept.

WS-Policy expressions are associated with various web services components using the WS-PolicyAttachment specification.

WS-Policy information can be embedded in a WSDL file, thus making it easy to expose web service policies through a UDDI registry.

### Web Services Security Policy (WS-SecurityPolicy)

WS-SecurityPolicy is part of the Web Services Secure Exchange (WS-SX) set of specifications hosted by OASIS (in addition to WS-SecurityPolicy, the WS-SX technical committee defines two other sets of specifications: WS-Trust and WS-SecureConversation, described later in this document).

WS-SecurityPolicy defines a set of security policy assertions used in the context of the WS-Policy framework. WS-SecurityPolicy assertions describe how messages are secured on a communication path.

Oracle has contributed to the OASIS WS-SX technical committee several practical security scenarios (a subset of which are implemented in the forthcoming Oracle Web Service Manager 11g). Each security scenario describes WS-SecurityPolicy policy expressions.

WS-SecurityPolicy scenarios describe examples of how to set up WS-SecurityPolicy policies for several security token types described in the WS-Security specification (supporting both WS-Security 1.0 and 1.1).

The subset of the WS-SecurityPolicy scenarios to be supported by Oracle Web Services Manager 11g represents the most common customer use cases. Each scenario has been tested in multiple-vendor WS-Security environments.

To illustrate WS-SecurityPolicy, let's use a scenario supported by Oracle Web Services Manager: UsernameToken with plain text password.

As mentioned earlier in this document, Username token is one of the security tokens specified by WS-Security. This specific scenario uses a policy that says that a requester must send a password in a Username token to a recipient who has authority to validate that token. The password is a default requirement for the WS-Security Username Token Profile 1.1.

This scenario is only recommended when confidentiality of the password is not an issue, such as a pre-production test scenario with dummy passwords.

```
<wsp:Policy>
  <sp:SupportingTokens>
    <wsp:Policy>
      <sp:UsernameToken/>
    </wsp:Policy>
  </sp:SupportingTokens>
</wsp:Policy>
```

An example of a message that conforms to the above stated policy is shown below.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="...">
  <soap:Header>
    <wsse:Security soap:mustUnderstand="1" xmlns:wsse="...">
      <wsse:UsernameToken>
        <wsse:Username>Marc</wsse:Username>
        <wsse:Password Type="http://docs.oasis.org...>
          XYZ
        </wsse:Password>
        <wsse:Nonce EncodingType="...#Base64Binary">qB...</wsse:Nonce>
        <wsu:Created>2008-01-02T00:01:03Z</wsu:Created>
      </wsse:UsernameToken>
```

```
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <Oracle xmlns=http://xmlsoap.org/Oracle>
      <text>EchoString</text>
    </Oracle>
  </soap:Body>
</soap:Envelope>
```

The example above contains a <Nonce> element and a <Created> timestamp, which, while optional, are recommended to improve security of requests against replay and other attacks. A nonce is a randomly generated (unique) number. The timestamp can be used to define the amount of time the security token is valid.

### **Web Services Trust (WS-Trust)**

In a message exchange using WS-Security only, it is assumed that both parties involved in the exchange have a prior agreement on which type of security tokens they must use for sharing security information.

However, there are cases where these parties don't have such an agreement, as a result trust must be established before exchanging messages. Trust between two parties exchanging SOAP / WS-Security-based messages is established by implementing the WS-Trust specification.

Essentially, WS-Trust enables security token interoperability using a security token service (STS) that allows one party to request a security token from a trusted authority (an STS enables security token exchange, token issuance, and token validation). For example, Company A using simple username / password credentials may request a SAML assertion from a trusted authority to do business with Company B, which is expecting SAML assertions as security credentials.

WS-Trust defines a request / response protocol: A client sends a <RequestSecurityToken> (RST) to the STS, the STS replies with a <RequestSecurityTokenResponse> (RSTR).

STS defines its security policies using WS-Policy and WS-SecurityPolicy. Policies are typically accessed using WS-MetadataExchange.

### **Web Services Secure Conversation (WS-SecureConversation)**

WS-SecureConversation plays the same role in message-level security that SSL plays at the transport level. WS-SecureConversation leverages other WS-\* specifications, in particular WS-Security and WS-Trust.

WS-SecureConversation defines the creation and sharing of security contexts between communicating parties and specifies how derived keys are computed and passed. A security context is an abstraction that refers to an established authentication state and negotiated keys. Security contexts mitigate the overhead involved in multiple-message exchanges. Security contexts are defined as a new WS-Security token type obtained using a binding of WS-Trust.

WS-SecureConversation defines a <wsc:SecurityContextToken> (SCT) element to support the requirements of security contexts (SCT is a “wire” representation of the security context concept). An SCT involves a shared secret used to sign and/or encrypt messages.

A security context needs to be created and shared by the communicating parties before being used. The WS-SecureConversation specification defines three ways of establishing security contexts:

- The security context is created by a Security Token Service (see WS-Trust).
- The security context is created by one of the communicating parties and propagated with a message leveraging the WS-Trust mechanism.
- The security context is created through negotiation and exchanges also leveraging the WS-Trust framework.

A security context token contains a shared secret using derived keys for signing and encrypting messages associated with the security context (a key can be derived from any security token that has a shared secret). Using a common secret, the communicating parties may define different key derivations. For example, four keys may be derived so that two parties can sign and encrypt using separate keys.

Different algorithms can be used to derive keys but the WS-SecureConversation specification defines its own algorithm based on a subset of the mechanism defined for TLS (see SSL above), using the P\_SHA-1 function to generate a sequence of bytes that can be used to generate security keys.

### **Web Services Federation (WS-Federation)**

WS-Federation provides support for the secure propagation (across Internet domains) of identity, attribute, authentication, and authorization information.

By relying on the models defined in WS-Security and WS-Trust, WS-Federation enables brokering of trust and security token exchange, support for privacy by hiding identity and attribute information, and federated sign-out. WS-Federation leverages WS-Policy and WS-MetadataExchange to describe and acquire metadata.

Oracle Identity Federation supports WS-Federation web (passive) requesters.

Finally, this section would not be complete without mentioning essential public key infrastructure (PKI) standards contributing to SOA security.

### **Public Key Cryptographic Standards (PKCS)**

PKCS is set of specifications developed and maintained by RSA Security (now part of EMC). There are currently 12 PKCS specifications. The most common are:

- PKCS#1 (RSA Cryptography Standard),
- PKCS#5 (Password-Based Cryptography Standard),
- PKCS#7 (Cryptographic Message Syntax Standard),

- PKCS#8 (Private-Key Information Syntax Standard),
- PKCS#10 (Certification Request Syntax Standard),
- PKCS#11 (Cryptographic Token Interface Standard),
- PKCS#12 (Personal Information Exchange Syntax Standard).

Oracle supports PKCS#1, PKCS#7, PKCS#8, and PKCS#11 in Oracle Security Developer Tools (OSDT), a set of Java-based cryptographic libraries used throughout Oracle Fusion Middleware components.

#### **Public-Key Infrastructure – X.509 (PKIX)**

PKIX is an Internet Engineering Task Force (IETF) working group. The goal of the PKIX work group is to develop Internet standards to facilitate the use of X.509-based PKI environments. These standards cover areas involving X.509 certificates, Certificate Revocation List (CRL), Lightweight Directory Access Protocol (LDAP), Certificate Management Protocol (CMP), Online Certificate Status Protocol (OCSP), Time-Stamp Protocol (TSP), and Cryptographic Message Syntax (CMS).

PKIX standards are supported by the Oracle Security Developer Tools (OSDT).

#### **XML Key Management Specification (XKMS)**

XKMS defines protocols for distributing and registering public keys. Applications or web services supporting XKMS don't have to deploy a PKI solution locally. The application or web service sends XML requests (in SOAP messages) to PKI components installed at a trusted third-party site (e.g., Verisign) which executes the XML requests on behalf of the requesting party (typically, XML requests are for issuing, retrieving, or revoking a certificate). XKMS works in conjunction with XML Encryption and XML Signature.

### **ORACLE'S PRODUCT STRATEGY**

Oracle takes a holistic approach to protecting SOA deployments, including an identity management infrastructure (Oracle Access Manager, Oracle Internet Directory, Oracle Identity Federation, Oracle Web Services Manager), development and deployment tools (Oracle Security Developer Tools (OSDT), JDeveloper (Java Integrated Development Environment), Application Development Framework (ADF), and a secure, governance-aware runtime environment (Oracle Components for Java – OC4J), including a UDDI registry).

As part of Oracle Fusion Middleware, Oracle provides an encompassing SOA security and management solution that allows companies to externalize security outside applications and web services, combine transport-level and application-level protection, and use a layered defense system (i.e., security gateways reside in the DMZ to ensure “perimeter” security, and local interceptors (or “agents”) provide “last-mile” security behind the inner network firewall).

All the key industry standards described above are supported by the various components of the Oracle solution.

## **CONCLUSION**

The increasing popularity of service-oriented architectures has introduced the need for additional standards to help support the new security challenges involved with this new paradigm.

Oracle has been instrumental in contributing to emerging standards, in particular the specifications hosted by the OASIS Web Services Secure Exchange technical committee.

In terms of identity management and service-oriented architectures security, Oracle's strategy is to focus on well established standards such as SAML, essential for identity federation, identity propagation, and end-to-end security from the user's web browser all the way across SOA-based transactions involving multiple web services.

Oracle Fusion Middleware components provide full support for key SOA security standards, enabling interoperability with heterogeneous, multiple-vendor security infrastructures.



Web Services Security

January 2008

Author: Marc Chanliau, Oracle Fusion Middleware Product Management

Oracle Corporation

World Headquarters

500 Oracle Parkway

Redwood Shores, CA 94065

U.S.A.

Worldwide Inquiries:

Phone: +1.650.506.7000

Fax: +1.650.506.7200

[oracle.com](http://oracle.com)

Copyright © 2008, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.