National Security Agency

Information Assurance Directorate

Net-Centric Enterprise Services (NCES) Profile of Web Service Security: Simple Object Access Protocol (SOAP) Message Security (WSSE)

02 MAY 2008

Prepared for the

Defense Information Systems Agency (DISA)

By the

National Security Agency

9800 Savage Road

Fort George G. Meade, MD 20755

Table of Contents

List of Figures

Scope

This profile covers the collective requirements for Simple Object Access Protocol (SOAP) Message Security to support digital signatures, encryption, and security tokens within the context of the network-centric enterprise services (NCES) information assurance (IA) subsystem. Implementations conforming to this profile are expected to conform to the specified base standards, as well as additional requirements imposed herein.

The scope of this profile is strictly limited to SOAP message security as applied to entities exchanging SOAP 1.1 messages. This profile provides guidance on message integrity and confidentiality. Implementations that conform to this profile are also expected to support related functionality that is subject to other information assurance (IA) profiles.

1 GENERAL DESCRIPTION

This profile was created to provide guidance on SOAP Message Security that can be used when building Web Services (WS) service offerings as part of a Service Oriented Architecture (SOA). Within a SOA, consumers, producers, services, messages, and products all have different requirements and vulnerabilities. Security threats are more complex and solutions more difficult than often believed. Examples of security vulnerabilities to be addressed for secure exchanges within a SOA construct include: Message integrity, confidentiality, falsified messages, principal spoofing, man in the middle, forged claims, replay of old requests, replay of old responses, reordering of request or responses, replay of request to other providers, message modification, denial of service, content-borne threats, fraud.

A Web service is defined as “a self-contained, modular application that can be described, published, located, and invoked over the Web.” [1] Systems interact with a Web service by exchanging messages with it. Although there is no requirement that the actual payload be encoded as an eXtensible Markup Language (XML) structure, XML has emerged as the de-facto encoding scheme to exchange data between two systems that could otherwise not communicate.

This profile is built upon an entire stack of technology standards, including: HTTP, HTTPS, XML, Web Service Security (WSS), XML Digital Signature Syntax and Processing (XMLDSig), SOAP, WS-Addressing, and SAML. Figure 1 illustrates these relationships.

[1] http://www-3.ibm.com/software/webservers/hostpublisher/library/publications/guide40/guide16.htm

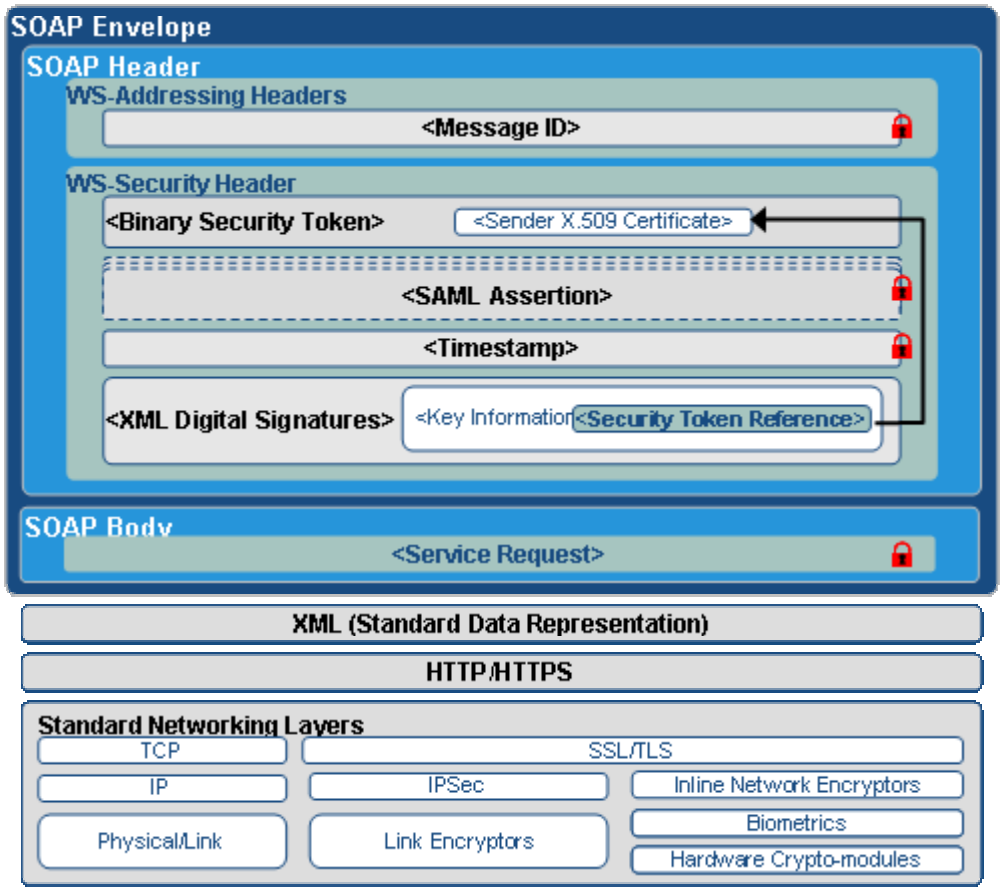

Figure 1 – Standards Underlying WSS Security Profile

1.1 Relationship to WSS

WSS defines how to apply the signing capabilities of XML Digital Signature (XMLDSig) to the body of a SOAP message. Finally, WSS addresses how to bind various authentication tokens, such as a userid/password pair, an X509 digital certificate, or even a Security Assertions Markup Language (SAML) Assertion, to a SOAP message and to use those authentication tokens. Rather than binding raw data packets to some standard network protocol, SOAP defines a standard message structure that is layered on top of an existing transport protocols to exchange messages with a Web service.

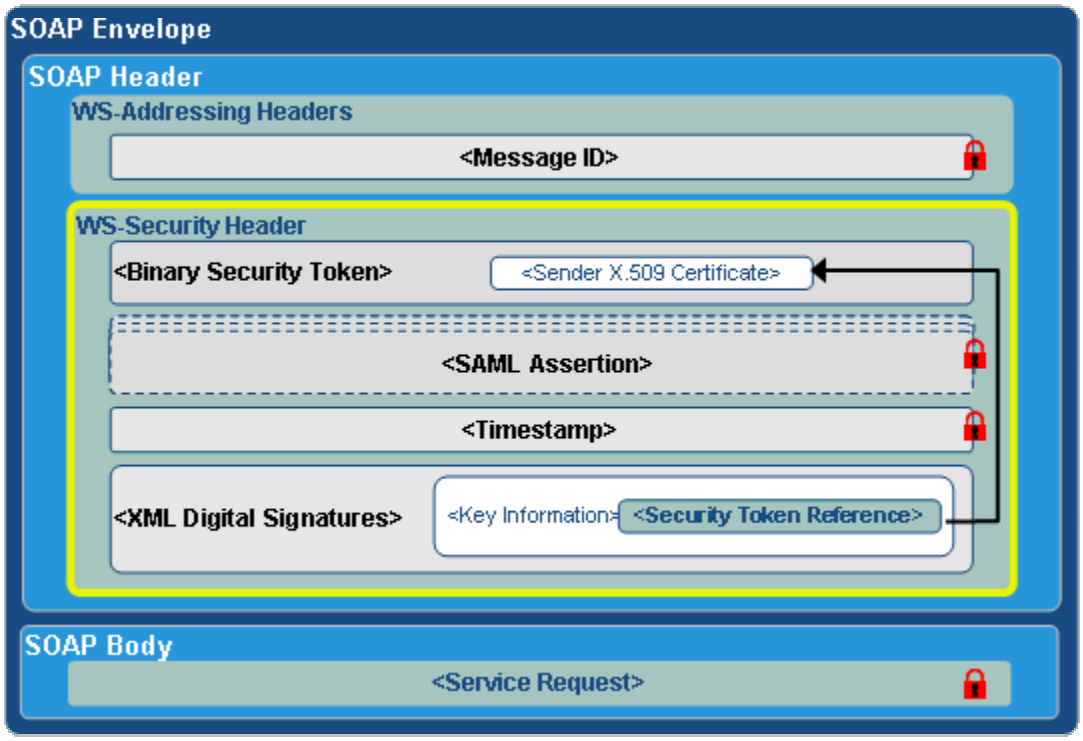

Figure 2 – Relationship to WSS

1.2 Relationship to SOAP

The Simple Object Access Protocol (SOAP1.1) is a specification from the World Wide Web Consortium, which defines a protocol for exchange of information in a decentralized, distributed environment. This definition describes an XML representation of a message, what is in a message and how to process a message. SOAP encourages vertical extensibility with the ability to introduce new pieces of information into a message as well as horizontal extensibility by targeting different parts of the same message to different recipients. At the top of the XML structure for a SOAP message, the root is represented as a <*SOAP:Envelope*> element. A <SOAP:*Body*> element wraps the data payload and any meta-data is wrapped in <*SOAP:Header*> elements. This generic envelope format provides a basis to exchange arbitrary and complex messages between collaborators independently of their internal infrastructures. There are multiple ways to secure a SOAP message. These can include some combination of securing the underlying communication infrastructure to attaching security tokens to the message itself. The Web Services Security (WSS) standard specifies how to implement security functions into SOAP messages by defining the syntax to encode message security and signature information directly within the SOAP message headers.

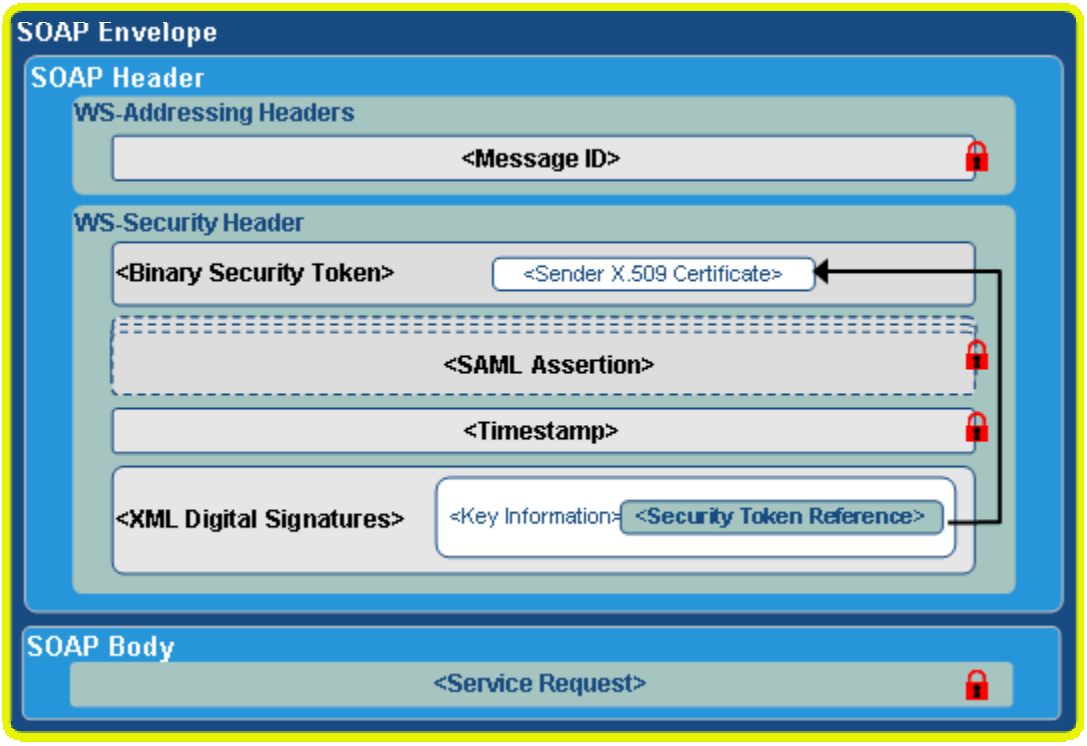

Figure 3 – Role of SOAP

1.3 Relationship to XML Digital Signature

This profile also uses XMLDSig, which can be found in XML Signatures Syntax and Processing Recommendation (XMLDSig) [XMLDSIG]. SOAP messages must be signed if the service provider requires authentication. The message signature provides message integrity, non-repudiation, and sender authentication. WSS relies on the XMLDSig to specify the syntax and processing rules for these within an XML context. XMLDSig also standardizes how to encode the resultant signature details into a native XML structure. Therefore, WSS syntax builds on XMLDSig syntax to standardize the representation of digital signatures within the <Header> elements of SOAP message. This profile specifies use of XML Digital Signature and WSS specifications to address the requirements of point-to-point and brokered message exchange patterns.

XMLDSig provides support for multiple signatures in that it provides the ability to sign specific portions of an XML construct. It permits different entities to sign distinct portions of a single document, preserving document integrity across multiple signatories. This capability is beneficial because if distinct portions of an XML document are signed, changes affect only the signatures on that specific portion of the data instead of all signatures. In addition, identical content generates the signed digest for each signature if multiple signatures are applied to the same portion of a document.

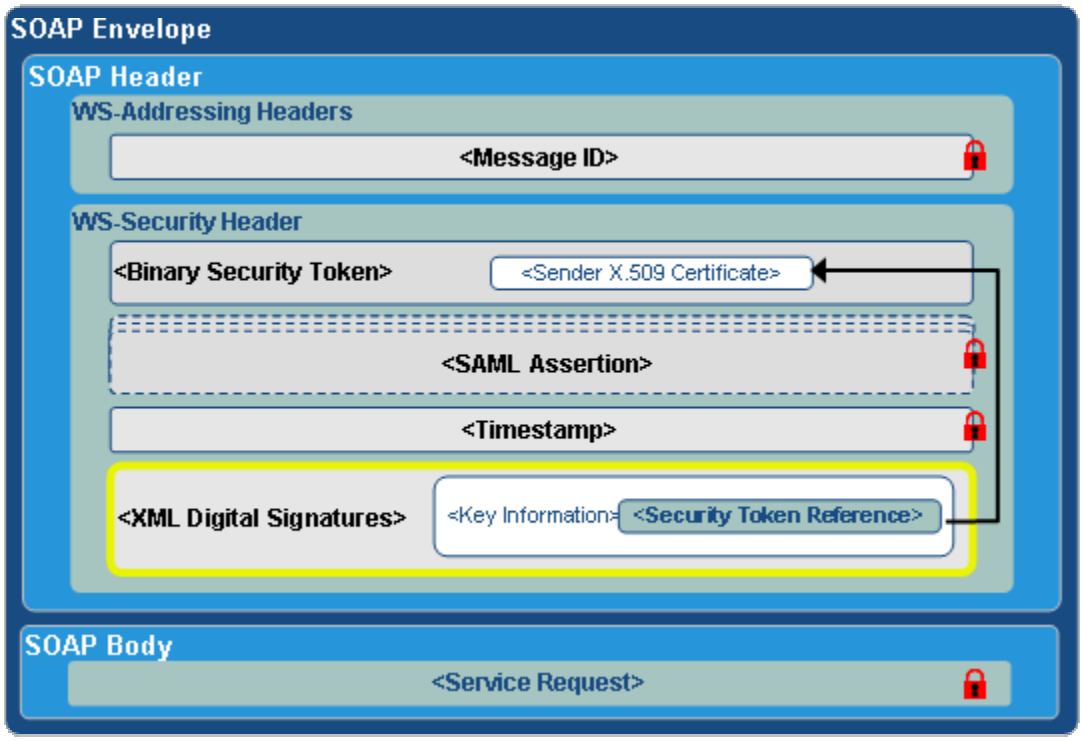

Figure 4 – Role of XML Digital Signature

1.4 Goals

The overarching goal of this profile is to build a baseline to provide a streamlined security solution that can be used across the enterprise. The following objectives derive from this goal.

- Provide guidance on SOAP Message Security for Web Services offerings.
- Provide guidelines for conducting secure SOAP message exchanges.
- Address known WS security vulnerabilities where possible through interface profiles.
- Capture requirements for SOAP v1.1 message security.
- Provide guidance on message integrity and confidentiality.
- Standardize security information attached to individual SOAP messages.
- Consolidate, profile, and apply many aspects of work done in other standards groups on WSS without defining new protocols.

This document does not define new protocols. It harmonizes existing work from open standards organizations to address the interoperability and security needs of a specific architecture concept. The initial version of this profile selected the NCES Security Services Architecture (v.5) as the baseline architecture concept. Additional information on this architecture concept, as related to this profile, can be found in Section 2.

1.5 Position within the Taxonomy

This specification is part of a profile taxonomy to augment existing web services (WS) standards to meet NCES Information Assurance (IA) requirements. The objectives of these profiles are to:

- Refine WS standards requirements to improve interoperability.
- Identify known gaps in the standards without necessarily taking actions on these gaps.
- Address known WS security vulnerabilities where possible through interface profiles.
- Harmonize standards profiles with existing security architecture efforts, in particular NCES.

The top-level taxonomy of major interface types is defined by the following three categories:

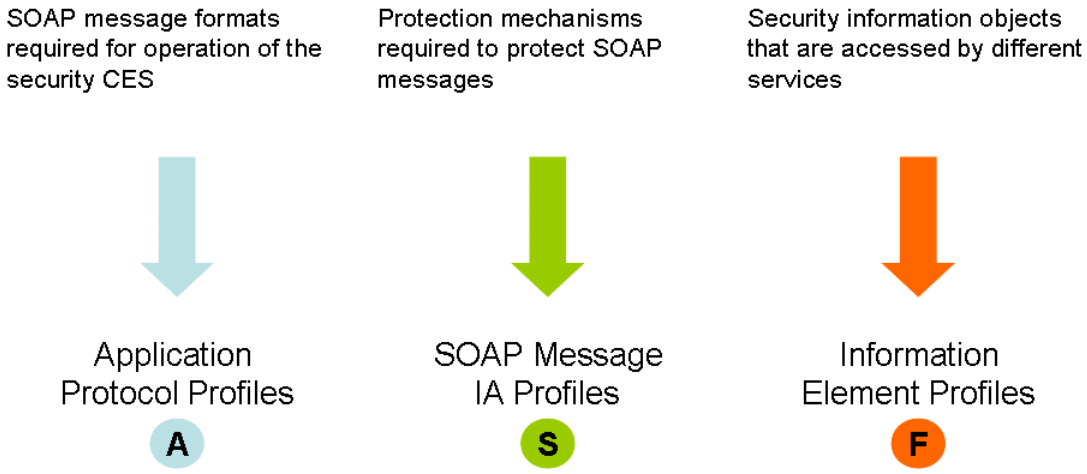

Figure 5 – Top Level Taxonomy of Major Interface Types

Specifically, this profile is part of the series of SOAP Message IA Profiles (i.e., S-profiles) that are described in the IA profiles taxonomy.[2] The scope of this specification addresses the following profile:

[2] Note that this taxonomy is not yet formally documented.

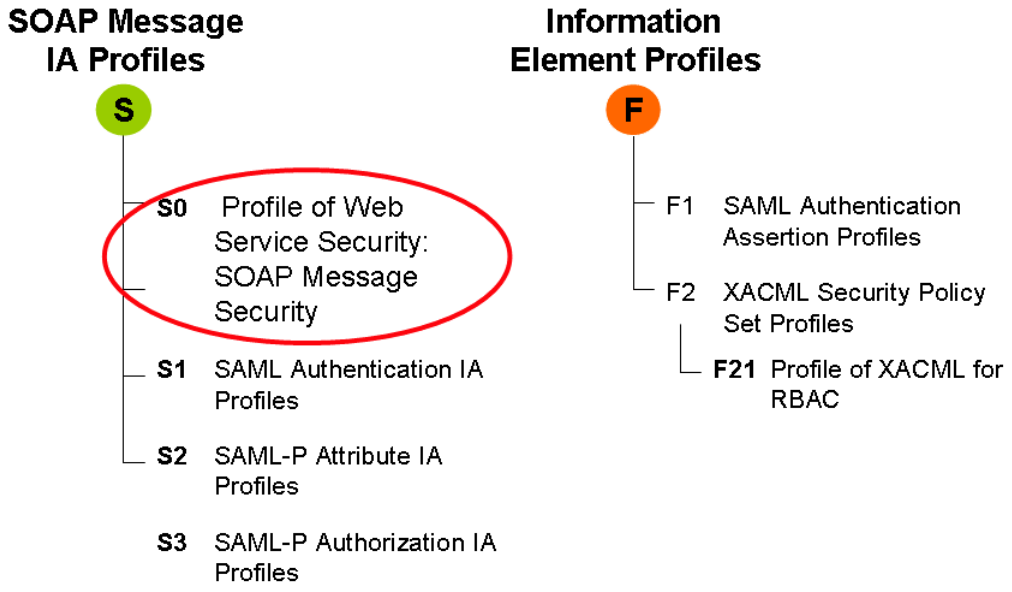

Figure 6 – SOAP IA and Information Element Profiles

1.6 Dependencies on other profiles

This section will contain information related to the dependencies between this profile and other profiles in the NCES IA profiles taxonomy.

1.7 Development Methodology

This profile first examined WS and security standards as well as typical security scenarios in a WS implementation of SOA. From this examination, point-to-point and brokered trust security scenarios were targeted. Next, identification and examination of existing documents from NCES, OASIS, W3C, NIST and IETF Standards, as well as the Web Services Interoperability Organization (WS-I) Basic Profile and Basic Security Profile, provided the basis for this profile. These documents include:

- NCES Service Security Design & Interface Specifications
- OASIS Web Services Security: SOAP Message Security 1.0
- XML-Signature and Syntax Processing
- XML-Encryption and Syntax Processing
- Web Services Addressing (WS-Addressing) Specification
- NIST documents were used as reference for specifying security requirements.

The security scenarios selected require a harmonization of existing standards. Therefore, interoperability issues may also arise from the interaction of combining the standards to

work within the architecture context. At times, the combination of standard options and solutions must be jointly (rather than individually) validated against the architecture concepts for each scenario. When combining standards, some of the options may not be available because the format and requirements of a standard may preclude the use of some options that are legal for another base standard. For example, applying XML Digital Signature to a SOAP message precludes the use of enveloping signatures so to remain compliant with SOAP protocol requirements.

2 PROFILE SCENARIO

As mentioned in Section 1.4, existing industry and community standards, such as WSDL, essentially provide building blocks with which to resolve these concerns. Standards represent consensus on needs, requirements and capabilities. These standards may originate from several sources such as International Standards (e.g., IETF), US National Standards (e.g., NIST), Federal or State Regulations (e.g., DoD Directives) and Consortia Specifications (e.g., W3C, OASIS) which identify the level of consensus backing the standard. This consensus forms a basis upon which multiple technologies can provide generalized functionality. However, a standard does not detail how to adopt, adapt and tailor this functionality to best fit the needs of a particular enterprise. Furthermore, individual standards cannot consider how to integrate the capabilities of a set of standards into the best solution for an enterprise. Architecture establishes the context and perspectives for which solutions are developed to address the needs of an enterprise. Therefore, architecture guides the evaluation of the problem space, identification of risks and eventual development of a solution.

The scenario for this profile addresses SOAP Message Security between a Web services consumer and a service provider as approached in the NCES Security Service architecture.

- In a Web Services environment, SOAP messages are the only medium of exchange between service providers and consumers.
- Service consumers and providers exchange security related information (e.g., certificates) with each other through open security standards such as WSS and SAML.
- Underlying security infrastructure capabilities (e.g. credential and policy management) are also exposed as Web Services using technology-agnostic WSDL interfaces.

This profile documents the critical architecture aspects of a Web Service based service-oriented system that cannot be captured exclusively through the specification of applicable Web Service standards. The solid lines in Figure 7 below indicate the applicability of this profile to the NCES Security Service Architecture.

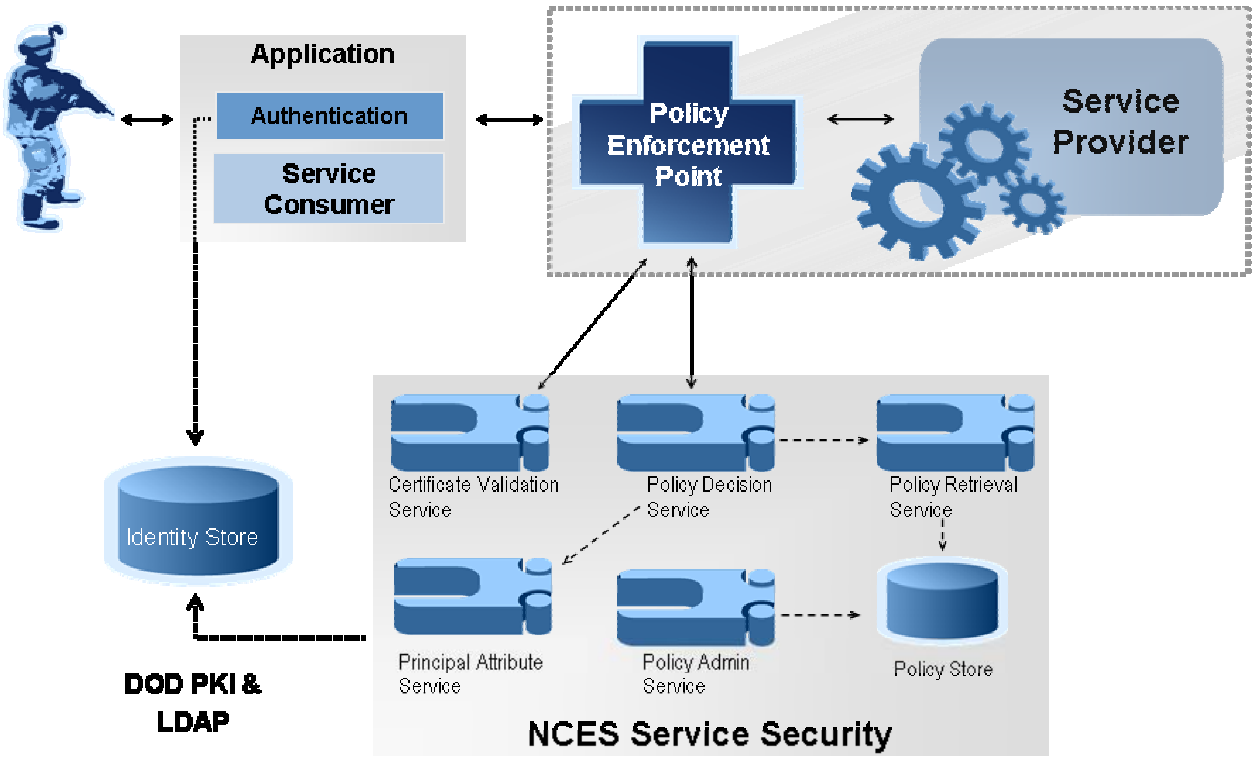

Figure 7 – Profile S0 Scenario

Service Security is comprised of infrastructure-level components that:

- Prevent unauthorized users from accessing Web Services.
- Enable enterprise security access policy to be set and enforced.
- Provide developers a mechanism to protect deployed service components.
- Clearly articulate the business processing rules necessary to enforce access to protected enterprise service components.
- Leverage existing industry standards and specifications from standards bodies such as OASIS and the W3C.

Figure 7 also shows these infrastructure components, which comprise Service Security specified by NCES. Note that the architecture concept does not expressly illustrate underlying communications infrastructures below the application layer.

This architecture relies on the following standards:

- WSS
  - Message Timestamp
  - Message signature PKI Certificate of Message Signer
  - Message Signature

- Optional SAML Assertions

- WS-Addressing
  - Unique message ID for every SOAP message
  - Note: Current NCES implementation follows vendor specification from IBM and Microsoft while waiting on final recommendation by W3C for additional adoption
- XML-DSIG
  - Provide SOAP Message integrity during a Web service transaction
  - Used within the context of WSS
  - Note: Implementations of canonicalization algorithms are still maturing which causes performance drawbacks

2.1 Point-to-Point Message Exchange Relationship

The point-to-point message relationship is a service invocation pattern in which the message sender is also the message originator. Each Web service request takes the form of a signed SOAP message. This message is sent directly from the originator to the target service provider, using standard Internet protocols (e.g., Hypertext Transfer Protocol (HTTP)). This case is depicted in Figure 8.

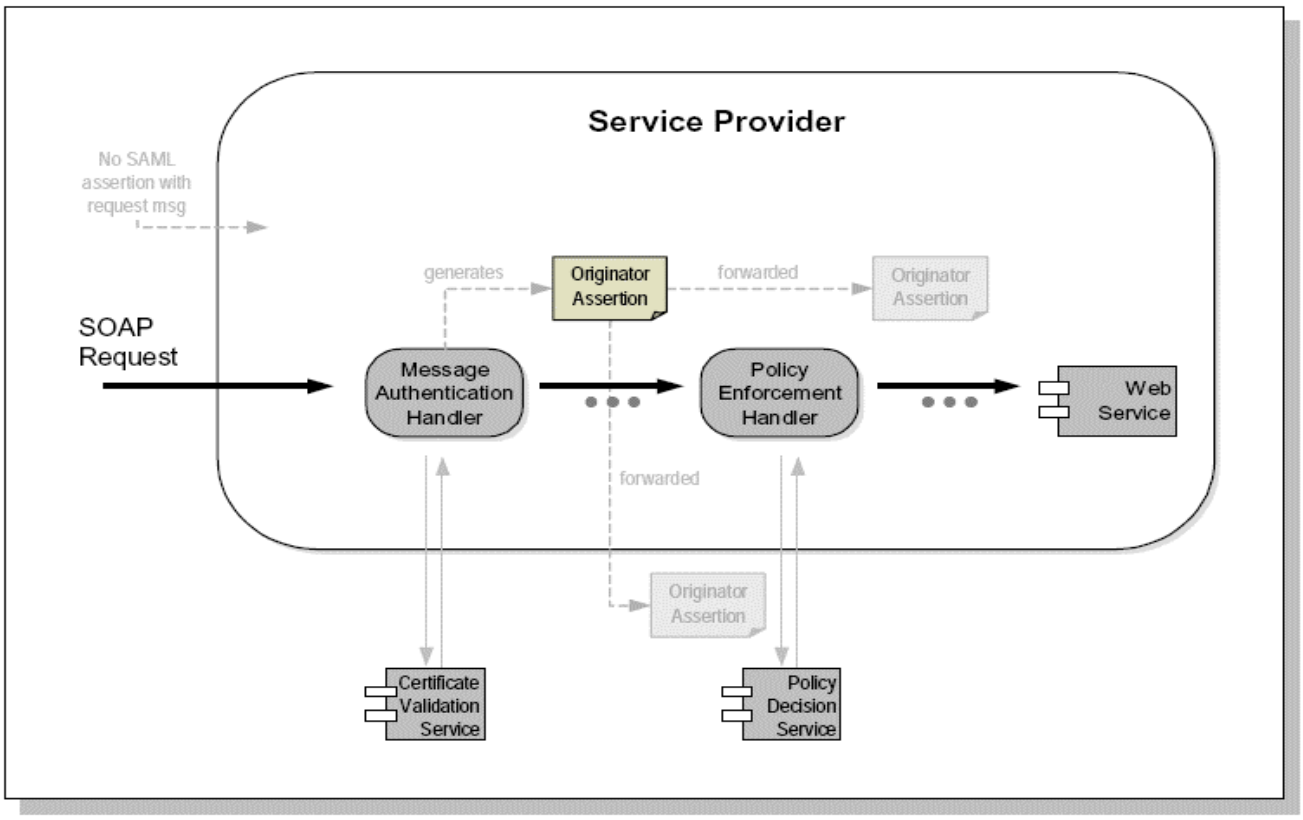

Figure 8 – Security in Point-to-Point Message Exchange Pattern

2.2 Brokered Message Relationship Exchange

The basic point-to-point invocation sequence may be easily extended to more complex usage scenarios. When service requests are chained, a Web service request may be issued on behalf of system entity other than the issuer itself. In this scenario, the message sender is not necessarily the message originator. For example, a service consumer requests a service from a particular provider (Provider A) which in turn requests a service of another provider (Provider B) on behalf of the original consumer. This would require Provider A to pass on the original security context (e.g. a SAML assertion) in the second SOAP request, as opposed to creating new one. Here Provider A "vouches for" the security context by signing the message with its digital signature, even though it is not the issuer of the assertion. In the same way that Provider A authorized the first request, Provider B then needs to authorize the second request based on both its policies applicable to the original consumer AND its trust relationship with Provider A.

Brokered message relationships may improve efficiency by performing certain authentication and authorization steps only once within a common security context. With a brokered message trust relationship, the message originator invokes the service of the intermediary to help establish a boundary for the unit of work so that the message originator is unable to forward a message directly to a target service provider even if the originator possess a valid user assertion. This use case is depicted in Figure 9.

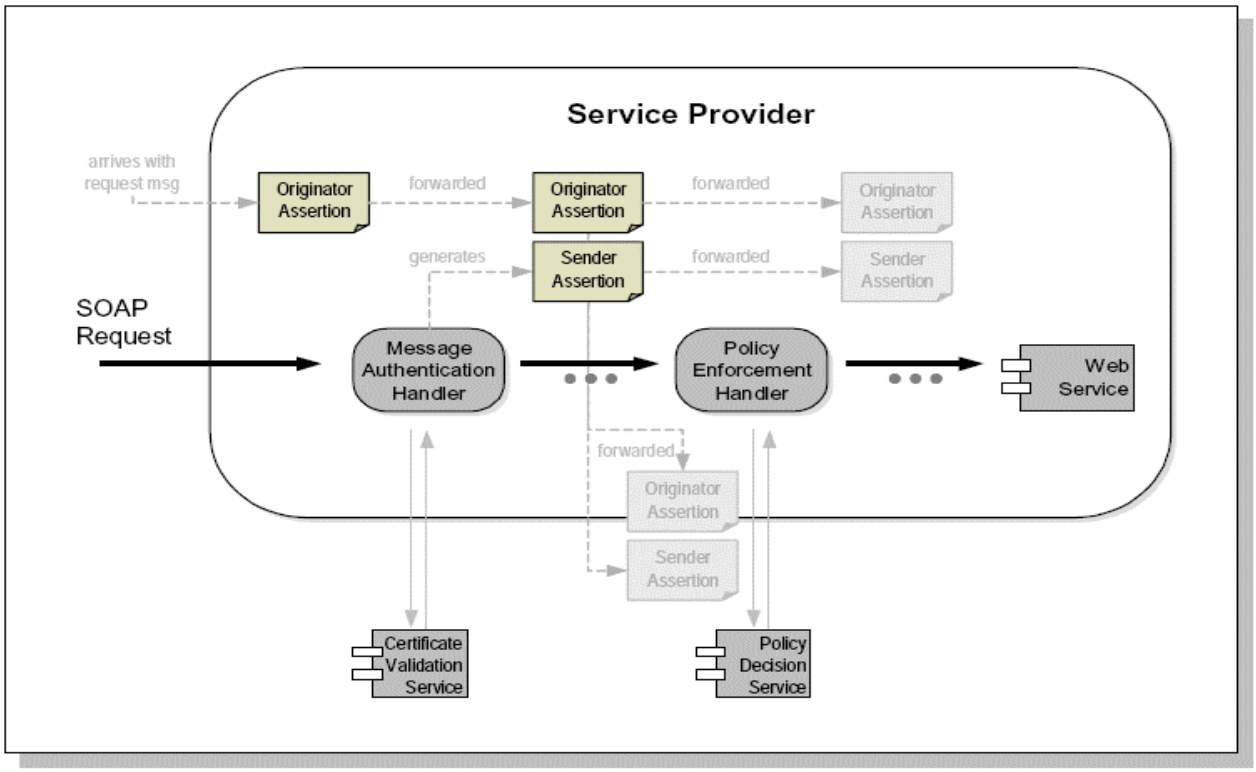

Figure 9 – Security in a Brokered Message Model

2.3 Assumptions

The following assumptions can be made for this profile:

- This profile applies to SOAP v1.1. It is anticipated that future versions of this profile will include support for SOAP v1.2. Currently, there is no binding between SOAP v1.2 and WSDL 1.1. Further, the following versioning rules between v1.1 and v1.2 are outlined in the SOAP v1.2 spec:

    1. When a SOAP 1.2 message reaches a SOAP 1.1 node it will generate a SOAP fault containing a version mismatch.
    2. When a SOAP 1.2 node receives a SOAP 1.1 message it can do one of the two things as stated below:
        - The node may process the SOAP 1.1 message.
        - It generates a Fault containing a Version Mismatch.
- This profile does not specify the mechanism used for edge authentication. However it is assumed that edge authentication was performed before the creation of the SOAP message.
- This profile assumes that the communication infrastructure is secured, using protocols such as secure sockets layer (SSL) and transport layer security (TLS) or secured infrastructure, in accordance with DOD policy.
- This profile does use SAML tokens. However, the profile is not using SAML authentication assertions to enable single sign-on. SAML authentication assertions are being used as tokens as part of the brokered message model to let the receiver know who the initial originator is and perhaps the identity of every intermediary in the chain.
- This profile uses of WS-Addressing, which is specified in Web Services Addressing Specification [WS-ADDR].

2.4 Data Structures

The following data structures are represented as XML elements:

- ***Secured Message***: A SOAP message that is secured in accordance with WSS and utilizing XML-DSIG for message signatures. SOAP messages are used for all Web Service communications.
- ***Authentication Assertion***: A SAML assertion that contains security context information related to the act of authentication. The Authentication Assertion contains a SAML AuthnStatement that indicates things such as the issuer of the assertion (presumably the

entity that performed the authentication), the date/time of authentication, the subject that was authenticated, and the method of authentication.

- ***Attribute Assertion***: A SAML assertion that contains security context information related to the attributes for an entity. The Attribute Assertion contains a SAML AttributeStatement that indicates things such as the issuer of the assertion (the attribute authority), the date/time of assertion, the subject of the assertion (corresponding to the subject from the request), and the attributes associated with the subject.

- ***Authorization Decision Assertion***: A SAML assertion that contains security context information relating to the result of an Authorization Policy evaluation. The Authorization Decision Assertion contains a SAML AuthzDecisionStatement that indicates things such as the issuer of the assertion (presumably the entity that performed the Authorization Policy evaluation), the date/time of the decision, the action and resource (corresponding to the request), and the decision result (e.g., PERMIT, DENY).

3 DEFINITIONS

This section establishes the conventions and definitions used in this profile. That includes specific terminology, general terminology, and the profile classification scheme.

In the context of the body of this profile, the requirements language of the Internet Engineering Task Force will be used. In this document the key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL are to be interpreted as described in IETF RFC 2119 [RFC2119].

The following general definitions apply to this profile:

Base Standards: References to the base standard or base standards in this profile imply the underlying standards cited as normative references in clause 2.1.

Basic Requirement: A protocol element, function, procedural element, or other identifiable feature specified in the base standards for which support is required by all implementations conforming to this profile.

Functional Group: A specification of one or more related protocol elements, functions, procedural elements or other identifiable features specified in the base standards that together support a significant optional area of functionality in this profile.

4 PROFILE REQUIREMENTS

WSS standardizes the representation of security information (e.g. signatures and tokens) contained within SOAP message headers. This profile version addresses message signatures. This profile also provides support for multiple signatures on a message. Future versions of this profile will address encryption at the message layer. Therefore, this version of profile assumes that the underlying communications infrastructure provides message confidentiality.

This profile provides a basic application of the WSS standard to the typical security scenarios encountered in a Web services implementation service-oriented architecture (SOA). The following sections address specific aspects of secured SOAP messages, such as digital signature representation and signature types, encryption types, tokens supported, HTTP headers, SOAP attributes, message identification and timestamps, and signature processing rules.

According to the requirements of this profile, SOAP requests MUST be signed if the service provider requires authentication. The digital signature provides the ability to verify message integrity and sender non-repudiation, but more importantly serves as the means for authenticating the sender of the message. The digital signature for the message can be verified using the designated signature, digest, and canonicalization algorithms (see Section 4.4).

Authenticating the message sender is crucial partly because the integrity of any embedded assertions depends on it. Fortunately, the WSS specification suite along with the XML-DSIG standard clearly defines the message signing syntax and semantics, which have been implemented in many existing commercial or open source toolkits. In this profile, asymmetric message signing and verification using DoD PKI certificates is supported.

4.1 User Identity

The scenarios supported by this profile require the user's identifier to be passed as security context information included in each service message (which is signed by the message sender). This identifier MUST be the user's X.509 distinguished name. When certificate-based authentication is employed, the user's distinguished name MUST be taken from the certificate used to authenticate the user. When another form of authentication is employed, it is the message sender's responsibility to obtain the corresponding X.509 distinguished name for inclusion in subsequent service calls. The message sender MUST also provide additional security context information such as when the authentication occurred and by which method.

4.2 HTTP Headers

Figure 10 illustrates where in the standards stack these requirements are addressed. Use of the SOAPAction header within an HTTP request containing a SOAP message was intended as a hint for the action to be performed on the body (i.e. SOAP message) of the request.

WS-I Basic Profile v1.1 clarifies the underlying specification regarding the SOAPAction HTTP header field. In a HTTP request message, the value of the SOAPAction HTTP header MUST be a quoted string. However, a SOAP processor MUST NOT depend on the value of the SOAPAction HTTP header. All vital information regarding the intent of a message is carried in soap:Envelope. According to the WS-I Basic Security Profile, the SOAPAction attribute of a soapbind:operation WSDL element SHOULD be either omitted, or have as its value an empty string. However, the WS-I Basic Security Profile does not impose requirements on the SOAPAction header itself. Therefore, the value for the SOAPAction header SHOULD contain the name of the operation to be invoked on the body of the message. Currently, it is RECOMMENDED that the name be a concatenated string with the following elements, in order:

- target namespace of the WSDL.

- the service name corresponding to the service element from the WSDL.
- the port name from the WSDL.

Finally, a SOAP processor MAY add arbitrary headers to the HTTP request but MUST NOT require any headers in the HTTP request to correctly process the message itself.

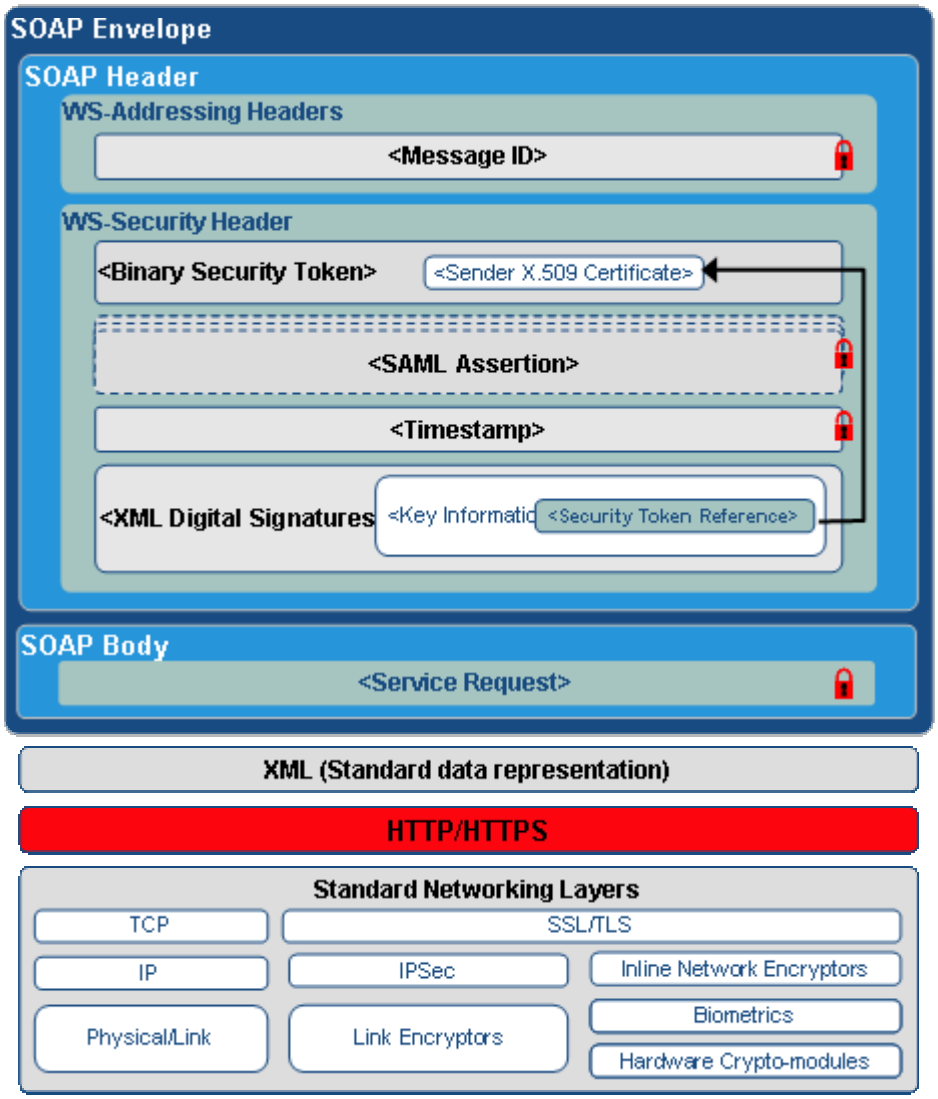

Figure 10 – Profile Requirements HTTP Headers

4.3 SOAP Attributes

Figure 11 illustrates where in the standards stack these requirements are addressed. The actor attributes from SOAP 1.1 SHOULD be used to specify which headers are intended for which intermediary. A value of "none" SHOULD indicate that the header is targeted to the ultimate

recipient of the message. Message security information targeted for different recipients MUST appear in different <wsse:Security> header blocks.

The S11:actor attribute[3] identifies a specific SOAP 1.1 actor. Although inclusion of this attribute is optional, no two-header blocks within a single message may omit an actor or specify the same actor.

All information targeted for a specific intermediary must be contained within a single header block. Therefore, two <*wsse:Security*> header blocks MUST NOT have the same value for S11:actor because of potential processing order issues (e.g., possible header reordering). The <*wsse:Security*> header block without a specified S11:actor MAY be processed by anyone but MUST NOT be removed prior to the final destination or endpoint.

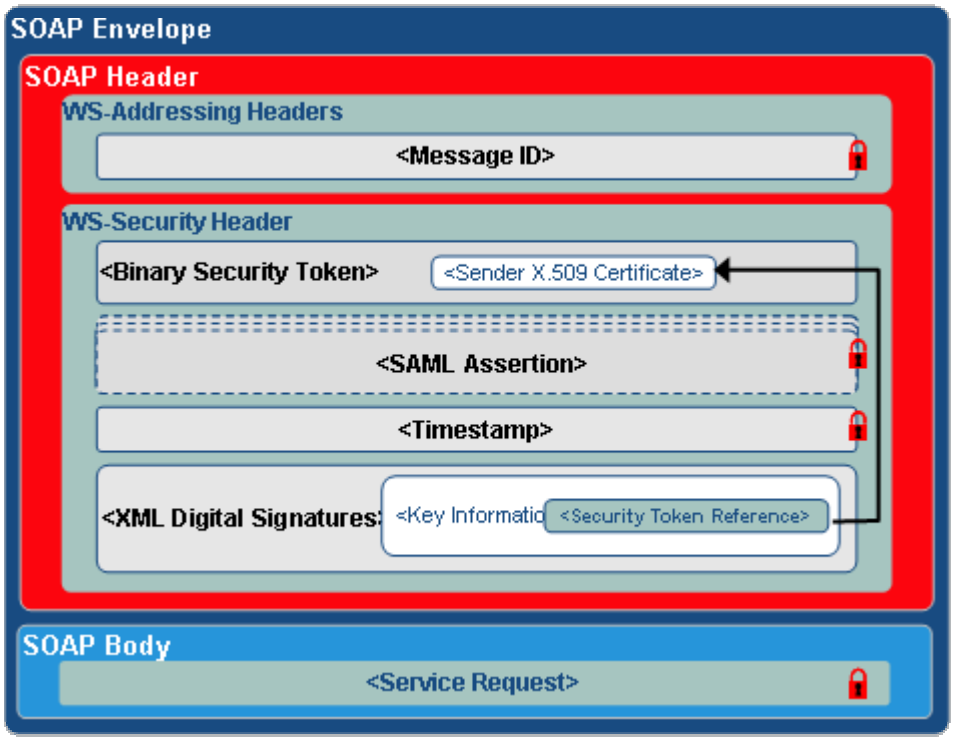

Figure 11 – Profile Requirements for SOAP Attributes

4.4 Digital Signatures

Figure 12 illustrates where in the standards stack digital signature requirements are addressed. WSS relies on XMLDSig to enable one or more signatures on a message. All signature information is represented in an XML fragment with a <*Signature*> element as the root. A <*SignedInfo*> child element reports details of the process that created the signature. A <*SignatureValue*> element contains the calculation of the signature over the information

[3] In SOAP 1.2, this attribute is now named role but retains identical semantics. Additional values for the role are also standardized.

represented in the <*SignedInfo*> element. Finally, an optional <*KeyInfo*> child element of the <*Signature*> *i*dentifies a key that may be used to verify the signature.

XMLDSig specifies three standard methods to associate a <Signature> with the signed content it secures: enveloping, enveloped, and detached. With an enveloping XML signature, the <Signature> construct also encapsulates the signed content. With an enveloped XML signature, the <Signature> element is inserted as a child element of the root of the signed XML document. A detached signature is completely separate from the signed data object, as it is over data external to the signature element. This profile narrows the scope of (XMLDSIG) by providing select options for securing SOAP messages.

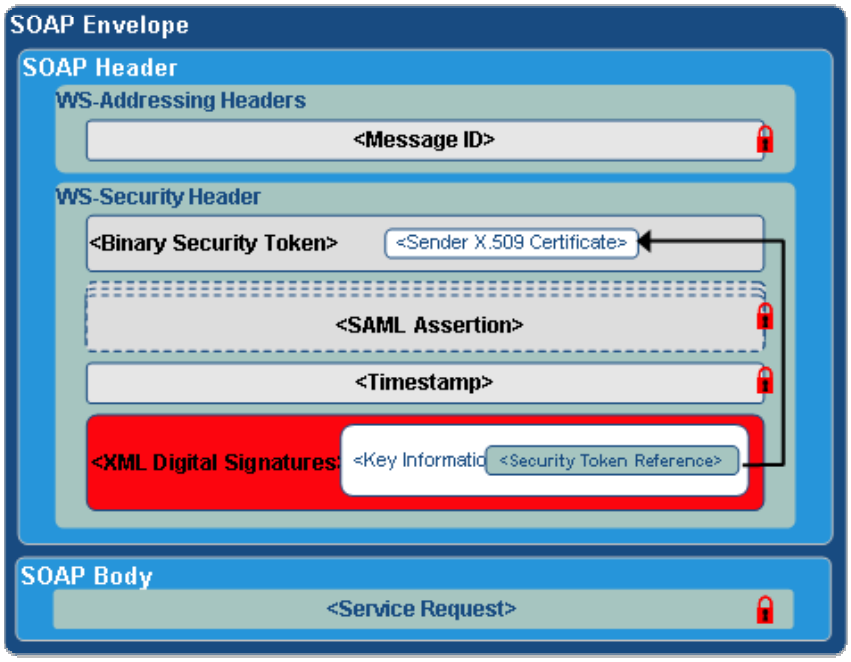

Figure 12 – Profile Requirements for XML Digital Signature

4.4.1 Signature Types

The XMLDSig specification identifies three standard methods to associate a signature with the signed content:

- Enveloping Signature – Not compatible for use with SOAP Message Security.
- Enveloped Signature – Approved for use with assertions.
- Detached Signature – Approved for use with SOAP Message Security.

The following sections detail these three methods of associating a signature with signed content.

4.4.1.1 Enveloping Signature

As depicted in Figure 13, an enveloping XML signature is one in which the construct containing all the information pertinent to the signing—such as the signature, signer identity, or Uniform Resource Identifier (URI) references to the signed data—encapsulates the signed content. For any SOAP-compliant message, however, a <*SOAP:Envelope*> must be the root element of the message. Therefore, *this signature method is not compatible with SOAP.*

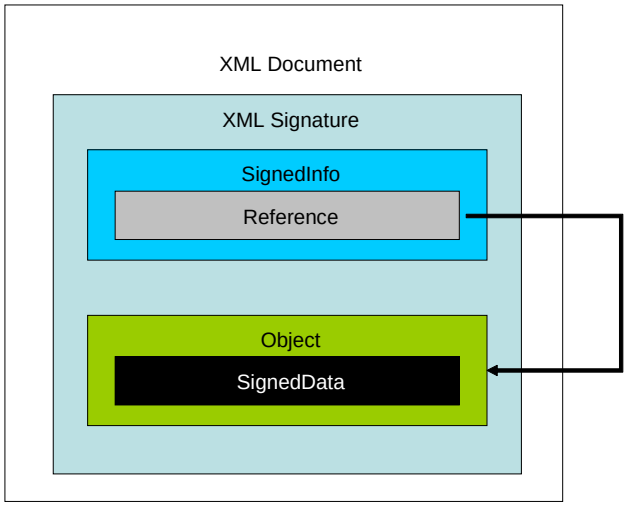

Figure 13 – Enveloping XML Signature

4.4.1.2 Enveloped Signature

As depicted in Figure 14, an enveloped XML signature is one in which the construct containing all the information pertinent to the signing—such as the signature, signer identity, or URI references to the signed data—becomes a child element of the root of the signed XML document. An *enveloped signature* comes from applying the *canonicalization* transform. This transformation excludes the signature element and its content from the digest generation. *The enveloped signature method, as defined by the XML Signature specification, is the preferred method to sign assertions attached to a SOAP message but MUST NOT be used to sign the SOAP message itself.*

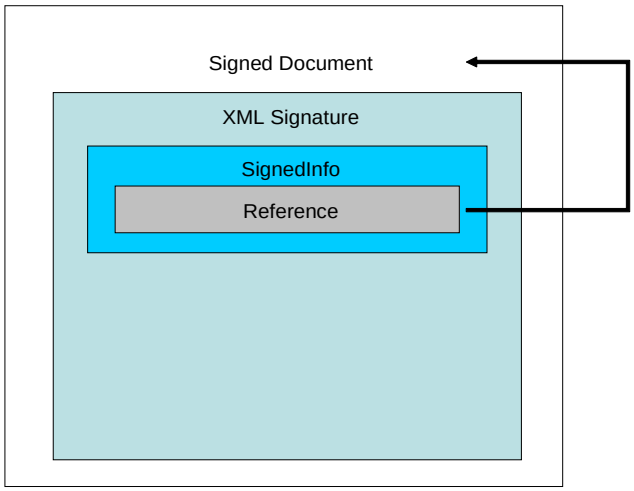

Figure 14 – Enveloped XML Signature

4.4.1.3 Detached Signature

As depicted in Figure 15, a detached signature is one in which the XML Signature construct containing all the information pertinent to the signing—such as the signature, signer identity, or URI references (Reference) to the signed content (SignedData)—remains independent from the signed content. In other words, the XML Signature construct is neither a parent (enveloping signature) nor child (enveloped signature) to the signed content. The Reference element may point to content contained within the same document, to content contained within an external resource, or to an external resource (signature applied across entire resource). A detached signature is useful when you can't modify the source data to be signed. The red circle with the line through it implies external detached references are not encouraged.

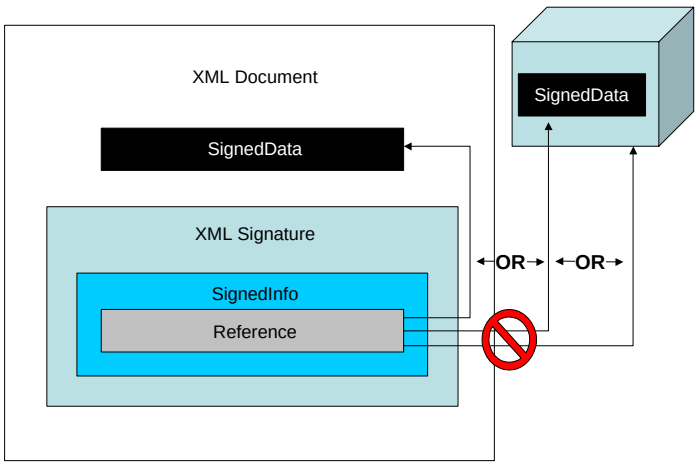

Figure 15 – Detached XML Signature

As depicted in Figure 16, the SOAP envelope structure provides a packaging format to unify two separate XML structures – the SOAP Body carrying the service request and the SOAP Header carrying the signatures. SOAP messages must be entirely self-contained. *The detached signature method MUST be used to sign the SOAP message body.*

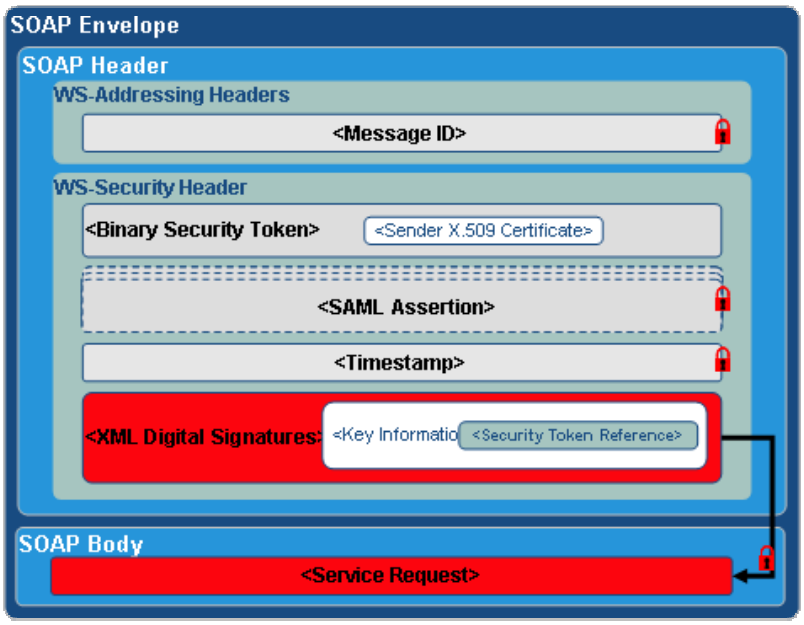

Figure 16 – Profile Requirements for Detached XML Signature with Single SOAP Structure

4.4.2 Further Restrictions

- URI attributes containing fragment identifiers MUST not be allowed. See [DSIG], Section 4.3.3.2
- Same-document XPointers MUST be supported. See [DSIG], Section 4.3.3.2
- The MgmtData child element of the KeyInfo element MUST NOT be allowed. See [DSIG], Section 4.4.7
- Signature applications MUST create, read, and verify XML content in Normalization Form C. See [DSIG], Section 7.0
- When processing, XML parsers must expand all non-character entities. See [DSIG], Section 7.1

4.5 Canonicalization and Transforms

Within the <*SignedInfo*> element, <*CanonicalizationMethod*> specifies how an XML resource is converted into an octet stream that will be palatable to the signature function. Unless proper canonicalization is performed, verification of signatures may not work because of changes to the elements in the containing scope. The Algorithm attribute in a signature MUST have a value of "http://www.w3.org/2001/10/xml-exc-c14n#" indicating Exclusive Canonicalization with or without comments. Exclusive Canonicalization with comments is preferred. If comments are removed before signing the data, recipients may be confused and assume the comments were not changed in transit, and hence trusted. Each <*Reference*> identifies individual pieces of content to be signed. Therefore, each reference MUST include at least one transform to specify the Exclusive C14N Canonicalization transform or a transform that itself incorporates Exclusive C14N Canonicalization, such as the *enveloped signature canonicalization* transform. This transform excludes the signature element and its content from the digest generation.

4.5.1 Supported Algorithms for Use with XMLDSig

This profile REQUIRES compliance with the core algorithm requirements of XMLDSig. This specification partitions algorithm compliance into the following categories, namely **Digest, Encoding, MAC, Signature, Canonicalization, and Transform.**

Compliant Implementations MAY support other algorithms described by RFC4051 but are NOT REQUIRED to do so.

4.5.1.1 Digest

Name	Identifier	Description
SHA1	http://www.w3.org/2000/09/xmldsig#sha1	(U)A SHA-1 digest is a 160-bit string. The content of the DigestValue element shall be the base64 encoding of this bit string viewed as a 20-octet stream.

Federal Information Processing Standard 180-2 [FIPS180-2], Secure Hash Standard, specifies four secure hash functions - SHA-1, SHA-256, SHA-384, and SHA-512 - for computing a condensed representation of electronic data (a message) in terms of the message digest length. When a message of any length $< 2^{64}$ bits (for SHA-1 and SHA-256) or $< 2^{128}$ bits (for SHA-384 and SHA-512) is input to a hash function, the result is an output called a message digest. The message digests range in length from 160 to 512 bits, depending on the hash function. The change notice specifies an additional hash function, SHA-224, may also be used.

4.5.1.2 Encoding

Name	Identifier	Description
base64	http://www.w3.org/2000/09/xmldsig#base64	The normative specification for base64 decoding transforms is [MIME].

4.5.1.3 MAC

Name	Identifier	Description
HMAC-SHA1	http://www.w3.org/2000/09/xmldsig#hmac-sha1	The HMAC algorithm (RFC2104 [HMAC]) takes the truncation length in bits as a parameter; if the parameter is not specified then all the bits of the hash are output.

From [FIPS198], HMAC shall be used in combination with an Approved cryptographic hash function. HMAC uses a secret key for the calculation and verification of the MACs. The size of the key, K, shall be equal to or greater than L/2, where L is the size of the hash function output. When a truncated HMAC is used, the t leftmost bytes of the HMAC computation shall be used as the MAC. The output length, t, shall be no less than four bytes (i.e., $4 < t < L$). However, t shall be at least $L/2 < t < L$ unless an application or protocol makes numerous trials impractical.

The default for HMAC-SHA1 complies with [FIPS198].

4.5.1.4 Signature

Name	Identifier	Description
DSAwithSHA1 (DSS)	http://www.w3.org/2000/09/xmldsig#dsa-sha1	DSA should be used in compliance with [FIPS186]
RSAwithSHA1	http://www.w3.org/2000/09/xmldsig#rsa-sha1	Refers to the RSASSA-PKCS1-v1_5 algorithm described in RFC 2437. The SignatureValue content for an RSA signature is the base64 [MIME] encoding of the octet string computed as per RFC 2437 , section 8.1.1.
ECDSA	http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha1	ECDSA is the elliptic curve analogue of the DSA. ECDSA is described in ANSI X9.62. It is RECOMMENDED that elliptic curves comply with Appendix 6 of [FIPS186].

From [FIPS186], the DSA algorithm takes no explicit parameters. The signature value consists of the base64 encoding of the concatenation of two octet-streams that respectively result from the octet-encoding of the values r and s in that order. Integer to octet-stream conversion must be done according to the I2OSP operation defined in the RFC 2437 [specification with an l parameter equal to 20].

From [FIPS186], the RSA digital signature algorithm is a FIPS approved cryptographic algorithm for digital signature generation and verification. This is described in ANSI X9.31. The FIPS change notice specifies that *n* should be at least 1024 bits.

4.5.1.5 Canonicalization

Name	Identifier	Description
Exclusive XML Canonicalization	http://www.w3.org/2001/10/xml-exc-c14n#	omits comments
Exclusive XML Canonicalization with Comments	http://www.w3.org/2001/10/xml-exc-c14n#WithComments	Preferred choice, see Section 4.5 in this document

The Exclusive XML C14N algorithms also take an optional explicit parameter of an empty InclusiveNamespaces element with a PrefixList attribute. The value of this attribute is a white space delimited list of namespace prefixes, and where #default indicates the default namespace, to be handled as per [XML-C14N].

4.5.1.6 Transforms

XSLT transforms MUST NOT be used. XSLT transforms allow for modification of the data to be signed. See Section 8.1.3 in [DSIG].

Application specific transforms MUST NOT be used. See Sections 4.3.3.4, 8.1, 8.1.1, 8.1.2, and 8.1.3 in [DSIG].

Name	Identifier	Description
Enveloped Signature	http://www.w3.org/2000/09/xmldsig#enveloped-signature	For use with assertions

4.6 Message Identification and Timestamps

Figure 17 illustrates where in the standards stack these requirements on Message Identification and Timestamps are addressed. Since even a valid, signed, SOAP message may be recorded and resent, or intercepted and resent at a later time, Section 13.2.1 of [WSS] states, “It is strongly RECOMMENDED that messages include digitally signed elements to allow message recipients to detect replays of the message....” The combination of the Message ID and timestamp of each message can be used to provide replay protection and detect a replay attack. The recipient of a message MUST check messages for potential replay.

Requirements for timestamps are as follows:

- The security header MUST contain a timestamp (i.e., <*wsu:Timestamp*>), as defined in the WSS specification.
- The timestamp MUST contain a <*wsu:Created*> element that records the message creation time relative to the sender’s clock.

- The timestamp MAY also contain a <*wsu:Expires*> element that represents the expiration of the message.
- All timestamps MUST be in the Coordinated Universal Time (UTC) format.
- The resolution of timestamps should extend to milliseconds.

If the timestamp has expired, the capability should exist to use the information stored in the cache to determine the age of the information. In this case, the MessageID of each message MUST be compared with the cached MessageIDs from previously received messages.

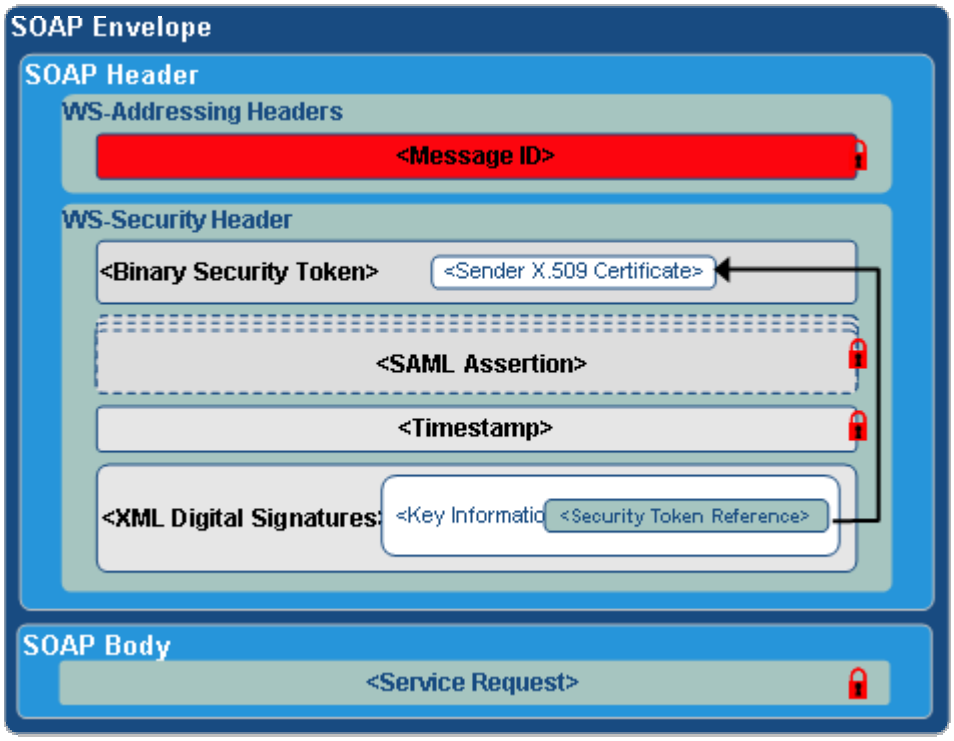

Figure 17 – Profile Requirements for Message Identifiers and Timestamps

4.7 Signed Content

Figure 18 illustrates which portions of the message must be referenced from within the digital signature to be compliant. Information from the edge authentication is made available as an SAML authentication assertion and is inserted into the <wsse:security> header. A message sender MUST NOT sign an assertion alone, but rather sign it along with other elements in the request message including the SOAP body. Signing the SAML assertion but not the request message would cause a serious security concern: Because there isn't a signature that cryptographically binds the assertion and the request body, the request body could be tampered with during transit. Further, the signed assertion could potentially be hijacked for other unintended uses. Signed or not, an assertion may be hijacked regardless (that is, when there is no message confidentiality), but a signed assertion might give recipients a false sense of

security. In addition, there must be a unique identifier in the form of a URI so that it can be located within the message during the signature generation process.

The signature MUST contain references to the following elements within the containing message using relative URI fragment[4]:

- The *SignatureMethod* MUST be RSA-SHA1.
- The *DigestMethod* MUST be SHA-1.
- The *CanonicalizationMethod* MUST be Exclusive Canonicalization.

The signature MUST contain references to the following elements within the containing message using relative URI fragment[4]:

- <wsa:MessageID>
- <saml:Assertion>, if present[5]
- <wsu:Timestamp>
- <soap:Body>

The signature MAY contain references to the following elements within the containing message using relative URI fragment[4]:

- <wsse:BinarySecurityToken> containing the message sender's digital certificate, if present.

[4] If only a fragment identifier is specified, then the reference is to the security token within the document whose local identifier (e.g., wsu:Id attribute) matches the fragment identifier

[5] The SAML Token profile for WSS does not describe the use of Direct or URI references to reference V1.1 SAML assertions

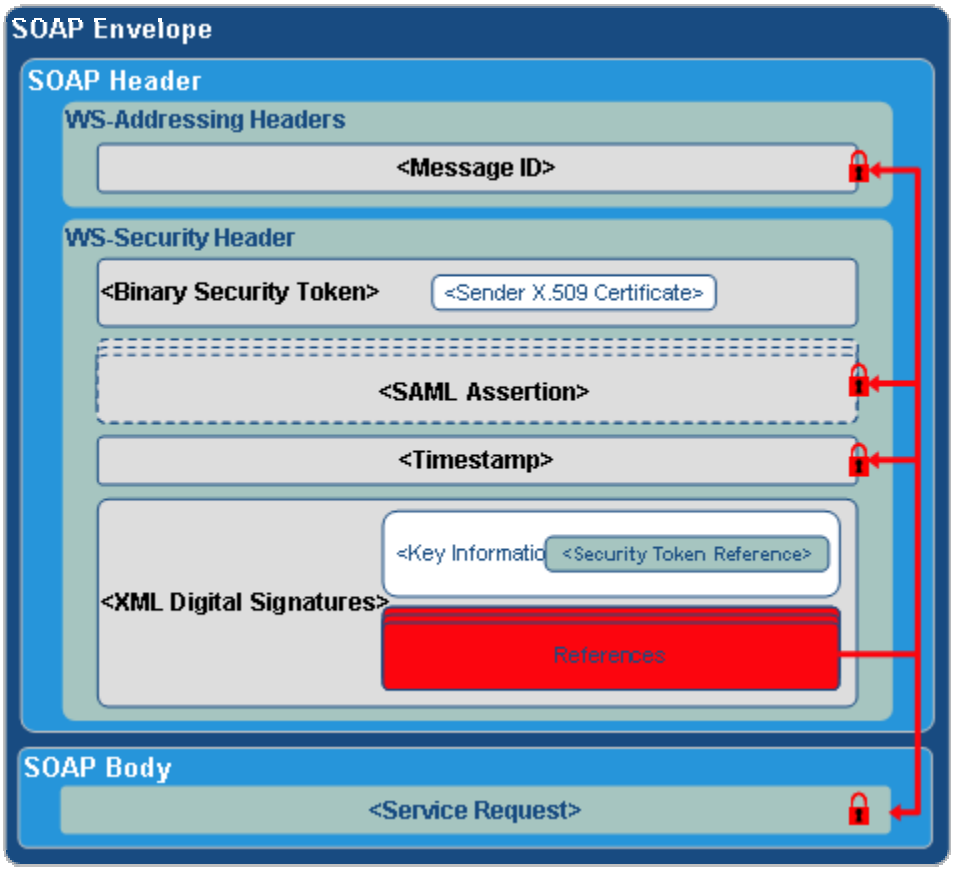

Figure 18 – Profile Requirements for Signed Content

4.8 Key Information Content

Figure 19 illustrates where in the standards stack these requirements are addressed. The <*ds:KeyInfo*> element of the signature MUST contain a <*wsse:SecurityTokenReference*>, which MUST contain either:

1. A <*ds:X509IssuerSerial*> reference (as defined in [X509]) that contains the Issuer Distinguished Name and Serial Number of the message sender's X.509 certificate

2. A reference to an embedded security token. In this case, the Security header MUST also contain a <*wsse:BinarySecurityToken*> that contains the message sender's X.509 certificate. The *ValueType* attribute of the binary token MUST be "*wsse:#X509v3*". In addition, the <*wsse:SecurityTokenReference*> itself MUST contain a <*wsse:Reference*> to the <*wsse:BinarySecurityToken*>.

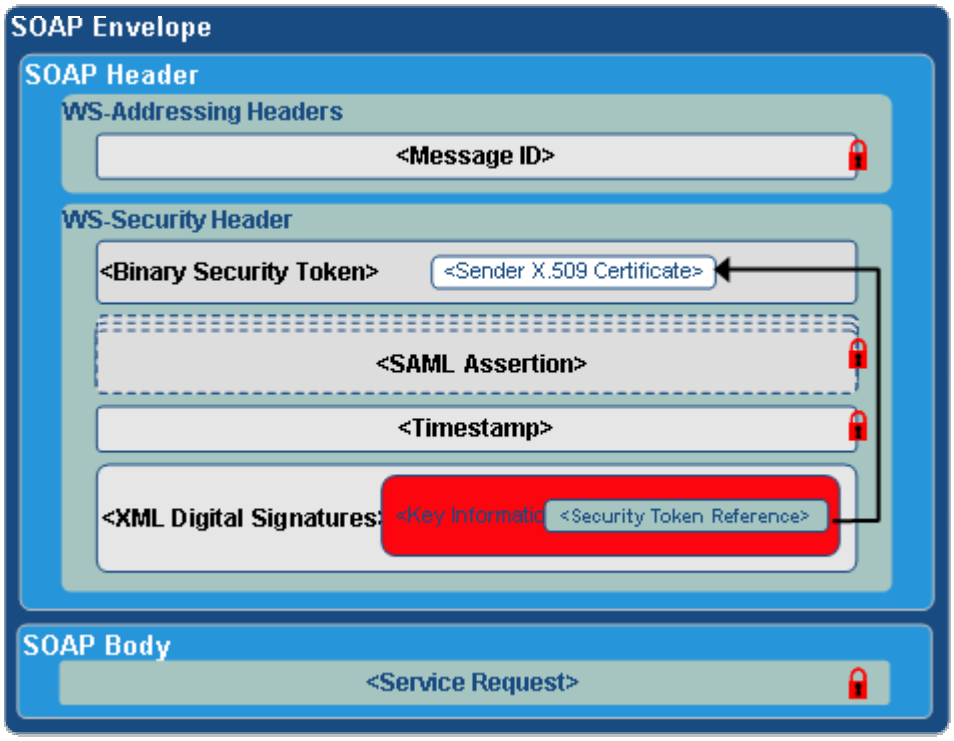

Figure 19 – Profile Requirements for Key Information Content

4.9 Token References

The <*ds:KeyInfo*> element of the signature MUST contain a <*wsse:SecurityTokenReference*>. This element MUST contain either a reference to an embedded security token or a <ds:X509IssuerSerial> reference. In the first case, the Security header MUST also contain a <wsse:BinarySecurityToken> that contains the message sender's X.509 certificate. In addition, the <wsse:SecurityTokenReference> itself MUST contain a <wsse:Reference> to the <wsse:BinarySecurityToken> as defined in [WSXCTP]. In the latter case, the <ds:X509IssuerSerial> reference (as defined in [WSXCTP]) MUST contain the Issuer Distinguished Name and Serial Number of the message sender's X.509 certificate.

4.10 Tokens Supported

WSS REQUIRES all SOAP implementations to be able to process Username and Binary security tokens. This profile DECOMMISSIONS functional support for the Username token in favor of the binary security token, which contains reference to the message senders Public Key Certificate. Further, this profile REQUIRES support for SAML assertions as attached tokens, and provides guidance for proper representation of SAML assertions within the <*wsse:Security*> construct. All Binary Security Tokens (i.e., ≤wsse:BinarySecurityToken>)must contain an X.509 certificate, as depicted in Figure 20. The ValueType attribute of the binary token MUST be "wsse:#X509v3".

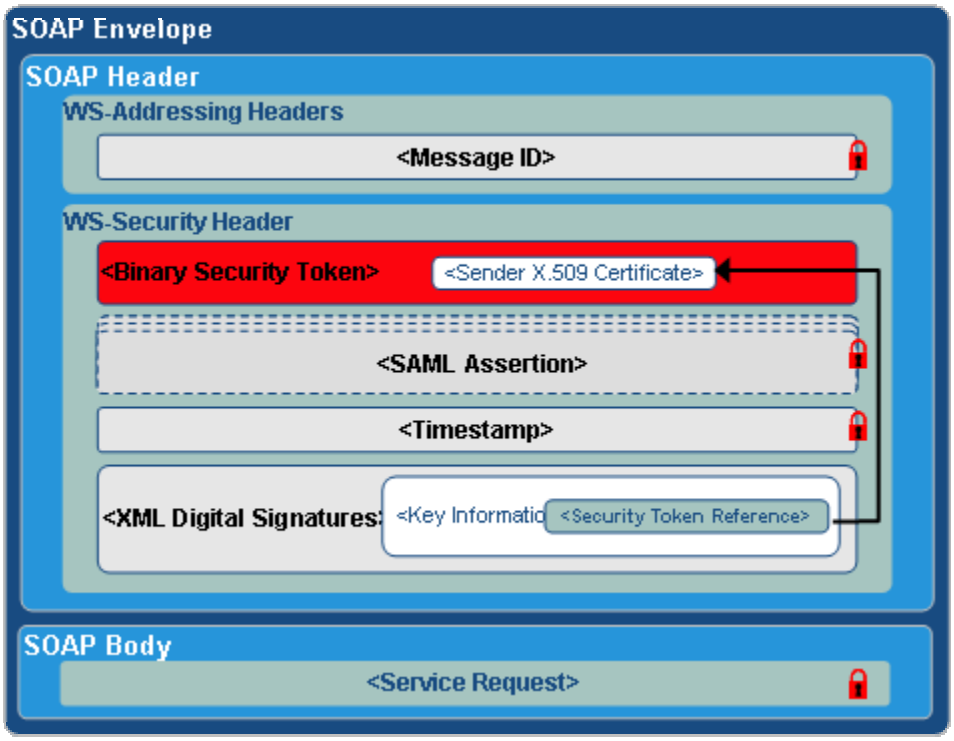

Figure 20 – Profile Requirements for Binary Security Tokens

When a <wsse:BinarySecurityToken> is referenced from a <ds:Signature> element, the canonicalization algorithm (e.g., Exclusive XML Canonicalization) should NOT allow unauthorized replacement of namespace prefixes of the QNames used in the attribute or element values. In particular, it is RECOMMENDED that these namespace prefixes be declared within the <wsse:BinarySecurityToken> element if this token does not carry the validating key. Although the full SAML assertion is beyond the scope of this profile, any implementation that is compliant with this profile SHOULD recognize an attached SAML Assertion as a child element of the <wsse:Security> header, as depicted in Figure 21. Any SAML assertion MUST use an X.509 Distinguished Name as a subject identifier within the assertion. Only SAMLAuthenticationStatements (SAMLAuthN) and SAMLAttributeStatements (SAMLAttrib) SHOULD be contained within any SAML assertion inserted as a child of the *<wsse:Security>* element. An assertion containing SAMLAuthN MAY provides security context information related to the act of authentication, such as the issuer of the assertion (presumably the entity that performed the authentication), the date/time of authentication, the subject that was authenticated, and the method of authentication. An assertion containing SAMLAttrib MAY convey attributes for an entity, including the issuer of the assertion (the attribute authority), the date/time of assertion, the subject of the assertion (corresponding to the subject from the request), and the actual attribute values appropriate to the subject.

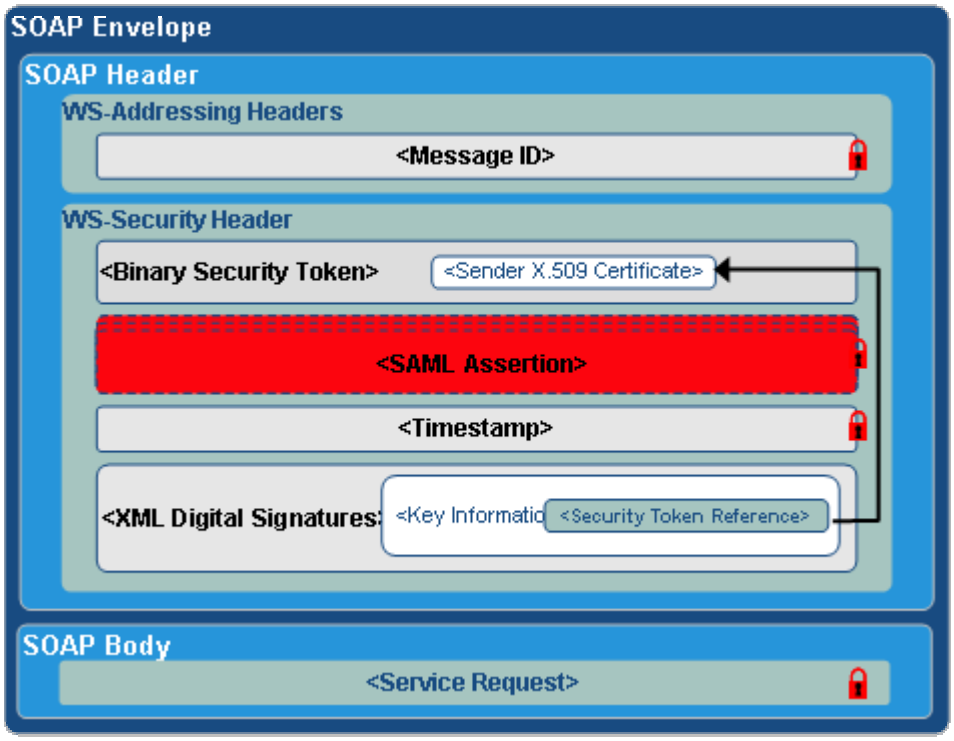

Figure 21 – Profile Requirements for Attached SAML Assertions

Whenever a SOAP message is sent on behalf of another principal, the Security header MUST contain the SAML assertion as defined above. An entity that receives a SOAP message without a SAML assertion within the<*wsse:Security*> header MUST assume that the sender of the message also is the originator.

4.11 Signature Processing Rules

The following rules apply to the message recipient:

- The signed elements specified by Section 4.6 MUST be verified against the signature, using the specified algorithms and transforms and the public key from the sender's certificate.

- The sender's X.509 certificate MUST be validated to ensure that:

  (1) It has not expired.

  (2) Its certificate validation path (or chain) can be validated to a trusted root authority.

  (3) Neither it nor any CA certificate in the validation path has been revoked (usually determined by checking the applicable Certificate Revocation Lists [CRL]).

- The MessageID of each message MUST be compared with the cached MessageIDs from previously received messages according to a configurable Freshness period.

This replay protection does not rely on trust in the sender's clock. However, freshness checking does require that system clocks remain somewhat synchronized across an enterprise to avoid self-imposed denial of service.

The following rules additionally apply for Request Messages:

- The end user assertion element, if found in the request message, MUST be validated against standard SAML processing rules.

- The issuer of the assertion SHOULD be checked against an "issuers list."

- When the message signature is validated, the handler MUST create a new SAML authentication assertion for the message sender.

4.12 Encryption Types

According to XML Encryption [XML-ENC], when encrypting arbitrary data (including entire XML documents), the EncryptedData element may become the root of a new XML document or become a child element in an application-chosen XML document. Enveloping Encryption, depicted in Figure 22, includes the cipher text within a cipher value block as a direct child element. Detached Encryption, which is shown in Figure 23, does not include a cipher value child element within the cipher data. Instead, it contains a cipher reference element that points to cipher text available elsewhere. When encrypting an XML element or element content the EncryptedData element replaces the element or content (respectively) in the encrypted version of the XML document. Although this profile does not currently REQUIRE encryption of data within a SOAP message, it is clear that Detached Encryption is not compatible with SOAP Message Exchange Patterns if the encrypted data is not contained elsewhere within the SOAP message. Therefore, any implementation that supports encryption MUST NOT select the Detached Encryption method.

This profile REQUIRES compliance with [FIPS197] which specifies the Rijndael algorithm (AES), a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits. AES was designed to handle additional block sizes and key lengths, but they are not FIPS compliant.

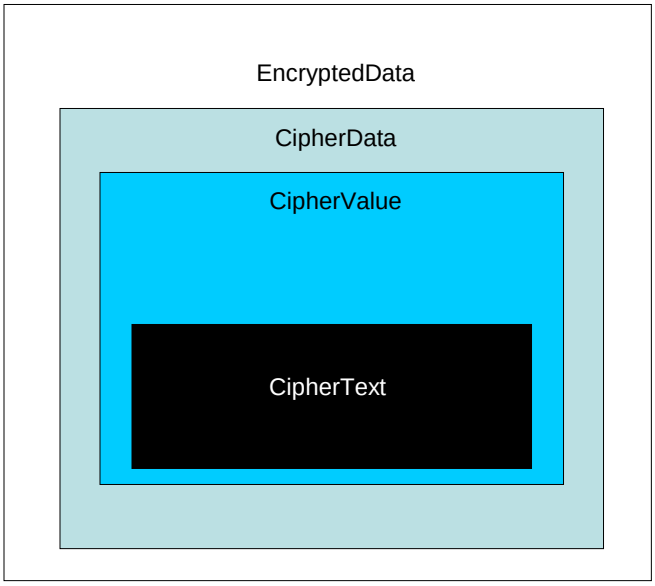

Figure 22 – Enveloped Encryption

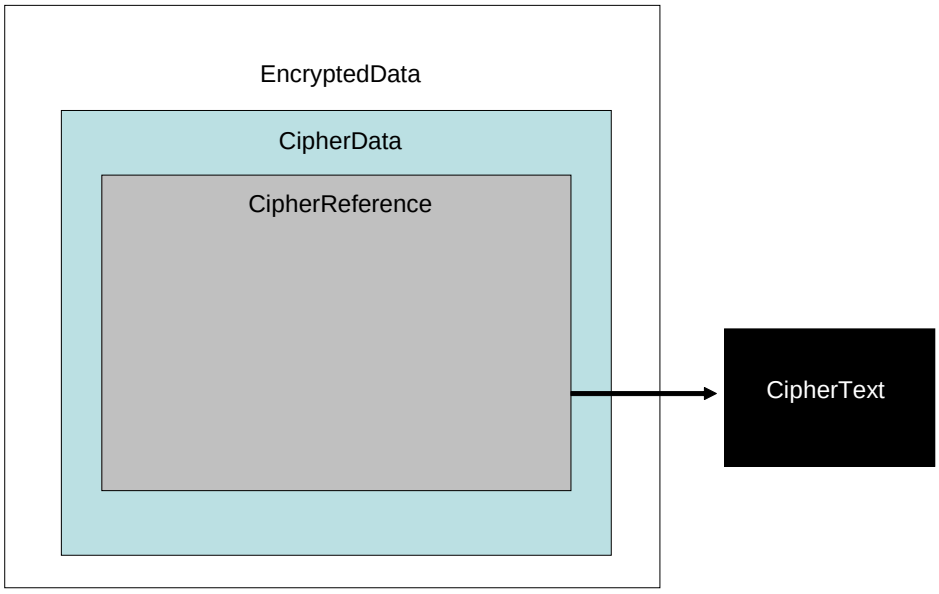

Figure 23 – Detached Encryption

5 CONFORMANCE

This profile enumerates several functions, elements, and so forth that have special applications; therefore, they are not required to be implemented in an implementation that claims to conform to the OASIS standard.

Additional conformance tests specific to this profile are still under development.

6 WAY AHEAD

This section will contain next steps and future directions of this profile as prioritized by the community. It is recommended that the prioritization of these future directions be determined by the risk assessments.

Examples of future profiles which may be closely related to this profile include:

- Profile for SAML Authentication Assertion used as attached tokens.
- Profile for signing and/or encrypting SOAP attachments.
- Relationships to SAML X.509 Attribute Sharing Profile and the WS-I SAML Token Profile.
- Encrypted SOAP messages.

7 REFERENCES

This section identifies external documents that are referenced by this profile.

7.1 Normative References

The following references form part of the basic foundation of this profile and must be supported by all implementations as qualified by the detailed profile requirements in Section 4.

[DSIG] *XML-Signature and Syntax Processing*, 12 February 2002, http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/ and http://www.ietf.org/rfc/rfc3275.txt

[FIPS180-2] http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf

[FIPS186] http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf

[FIPS197] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[FIPS198] http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf

[MIME] RFC 2045, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. N. Freed & N. Borenstein. November 1996. http://www.ietf.org/rfc/rfc2045.txt

[PASS] *OASIS Web Services Security: Username Token Profile 1.0*, March 2004, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

[WSA] Web Services Addressing (WS-Addressing) Specification, August 2004, http://www-106.ibm.com/developerworks/webservices/library/ws-add/

[WSSE] *OASIS Web Services Security: SOAP Message Security 1.0*, April 2004, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[XENC] *XML-Encryption and Syntax Processing*, 10 December 2002, http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/

[X509] *OASIS Web Services Security: X.509 Certificate Profile*, March 2004, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf

[XML-C14N] http://www.rfc-archive.org/getrfc.php?rfc=3741

[SOAP 1.1] http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[SOAP 1.2] http://www.w3.org/TR/soap12-part1/

7.2 Informative References

(U//FOUO) The following references provide additional information on related concepts that contribute to a fuller understanding of the context of this profile or the rationale for its requirements.

[NCESarch] (U//FOUO) *Net-Centric Enterprise Services (NCES) Security Core Enterprise Services (CES) Architecture*, Version 0.4 (Pilot), 2004 March 26.

[NCESspec] (U//FOUO) *Net-Centric Enterprise Services (NCES) Service Security Design Specification*, nces-soaf-ss-designspec-2005v01-WD-01, 2005 May 23.

[NCESintf] (U//FOUO) *Net-Centric Enterprise Services (NCES) Service Security Interface Specification*, nces-soaf-ss-interfacespec-2005v01-WD-01, 2005 May 23.

[NIST] *Guide to Secure Web Services: http://csrc.nist.gov/publications/nistpubs/800-95/SP800-95.pdf*

[RFC2119] *Key Words for Use in RFCs to Indicate Requirement Levels,* IETF RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC4051] *IETF RFC 4051 Additional XML Security URIs*, April 2005, http://www.ietf.org/rfc/rfc4051.txt

[WSI-BSP] *WS-I Basic Security Profile, current working draft available at:* http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0-2006-03-29.html

8 ABBREVIATIONS AND ACRONYMS

The following abbreviations and acronyms are applicable to this profile.

AES	Advanced Encryption Standard
CES	Core Enterprise Services
CRL	Certificate Revocation List
DISN	Defense Information Systems Network
DoD	Department of Defense
DSA	Digital Signature Algorithm
ECDSA	Elliptical Curve Digital Signature Standard
GIG	Global Information Grid
HMAC	Hashed Message Authentication Check
IETF	Internet Engineering Task Force
LTPA	Lightweight Third Party Authentication
MAC	Message Authentication Check
MD	Message Digest
NCES	Network Centric Enterprise Services
NIPRNET	Non-classified IP Router Network
NIST	National Institute of Standards & Technology
OASIS	Organization for the Advancement of Structured Information Standards
PGP	Pretty Good Privacy
PKC	Public Key Certificate
PKCS	Public Key Certificate Standard
PKI	Public Key Infrastructure
RACE	Research and Development in Advanced Communications Technologies in Europe
RIPEMD	RACE Integrity and Primitives Evaluation Message Digest
RSA	Rivest Shamir and Adleman
SAML	Security Assertion Markup Language
SHA	Secure Hash Algorithm
SKI	Subject Key Identifier
SIPRNET	Secret IP Router Network
SOAP	Simple Object Access Protocol
SPKI	Simple Public Key Infrastructure
URI	Universal Resource Identifier
WS	Web Services
WSDL	Web Services Description Language
WSS	Web Services Security
W3C	World Wide Web Consortium
XCBF	XML Common Biometric Format
XML	Extensible Markup Language
XrML	Extensible Rights Markup Language

9 OTHER CONSIDERATIONS

9.1 Relationship to XCCDF

The Extensible Configuration Checklist Description Format (XCCDF) specification defines a means for expressing security benchmarks in a way that fosters development of interoperable tools and content designed to permit the same document in multiple roles. XCCDF will be used predominantly at the infrastructure level, which is the major audience of these profiles.

The XCCDF specification has the possibility to work well with SOAP and XACML profiles. Moving forward, the XCCDF specification should be treated as either a framework or a template upon which to base all initial profiles. Each profile will remain flexible according to each scenario. XCCDF also profiles the framework to discuss profile requirements within the context of the Vulnerability Assessments. Finally, XCCDF may fit well in determining how secure policies are in XACML.

9.2 Relationship to NIST SP 800-95

The recent NIST publication, SP 800-95, **Guide to Secure Web Services** "describes how to implement security mechanisms in Web services. It also discusses how to make Web services robust against the attacks to which they are subject, [and summarizes] security techniques for Web Services." Therefore, future versions of this profile document will evaluate the requirements contained within the NIST SP 800-95 and align these requirements accordingly.

Annex A: Profile Tables

The tables in this appendix outline the requirements and restrictions on the individual elements, their values and their arguments as defined by the base standards. As consistent with the hierarchical nature of XML, the classification of information elements is relative to that of the containing information element, if any.

Where the children of an element are not individually specified, then each shall be considered to have the classification of that parent element.

Where the range of values to be supported for an element is not specified, then all values defined in the applicable base standard shall be supported.

To specify the support level of arguments, results and other protocol features for this profile, standardized terminology is used to classify each element – according to static and dynamic behavior.

Static Classifications

Static classifications describe the level of requirements for implementations to support the capability to use a particular feature of the protocol. The following classifications are used in this profile to specify static conformance requirements.

Mandatory Support (m): The element or feature shall be supported. An implementation shall be able to generate the element and/or receive the element and perform all associated procedures (i.e., implying the ability to handle both the syntax and semantics of the element) as relevant, as specified in the appropriate base standard. Where support for origination (generation) and reception are not distinguished, both capabilities shall be assumed.

Optional Support (o): An implementation is not required to support the element. If support is claimed, the element shall be treated as if it were specified as mandatory support. If support for origination is not claimed, the element is not generated. If support for reception is not claimed, an implementation may ignore the element on delivery, but will not treat it as an error.

Conditional Support (c): The element shall be supported only under the conditions specified in the profile. If these conditions are met, the element shall be treated as if it were specified as mandatory support. If these conditions are not met, the element shall be treated as if it were specified as optional support (unless otherwise stated).

Out of Scope (i): The element is outside the scope of the profile and will not be the subject of a conformance test or the element is not applicable in the particular context in which this classification is used.

Dynamic Behavior

Dynamic classifications describe the level of requirements for the behavior of implementations with respect to a particular feature. Dynamic conformance requirements are normally as specified in the applicable base standard. In some cases, however, it is necessary to specify additional dynamic conformance requirements in this profile. These are specified using a second classification code for particular elements. If no dynamic classification code is applied to an element, the required behavior is as specified in the applicable base standard. The following classifications are used in this profile to specify dynamic conformance requirements.

Required Use (r): The element shall always be present. An implementation shall ensure that the element is always generated or otherwise used, as appropriate. Absence of the element on reception shall result in termination or rejection of the communication with an appropriate error indication as specified in the base standards.

Prohibited Use (x): The element shall not be originated by an implementation claiming conformance to this profile. If the element is received it may be treated as a protocol violation unless otherwise stated.

When the requirements of this profile deviate from the requirements of the base standard, the rationale column provides motivating information for this difference. This rationale traces to the context and discussion contained within the main body of this profile document.

Global References and Remarks Notes:

C1 – This element is not present in v1.0 or v1.1 of the wsse, dsig, xenc, X509, or wsu specification.

C2 – This element is no longer part of the cited wsse, dsig, xenc, X509, or wsu base standards.

Table Column Descriptions:

Ref – Unique identifier assigned to this profile requirement to facilitate references to specific requirements in other profiles

XML Element – Namespace and name of element from base standard

Base Standard Usage – Allowable usage of XML element according to base standard requirements

Base Standard Rationale – Reasoning for expected usage of XML element according to base standard

Profile Usage – Allowable usage of XML element according to the requirements of this profile

Profile Rationale – Summary of reasoning for expected usage of XML element according to this profile

Reference and Remarks – Additional information related to this element

Namespaces Prefixes:

S11 – **http://schemas.xmlsoap.org/soap/envelope**

S12 – **http://www.w3.org/2003/05/soap-envelope**

wsse – http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

dsig – **http://www.w3.org/2000/09/xmldsig**

xenc – **http://www.w3.org/2001/04/xmlenc#**

wsu - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

Table A.1 Basic Requirements

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
1	<S11:Envelope>	mr	Section 4 of SOAP 1.1 indicates this element is required.	m	See Table A.2
2	<S12:Envelope>	mr	Section 5 of SOAP 1.2 indicates this element is required.	i	See Table A.3 and Section 2.3 Assumptions.

Table A.2 S11 Envelope

Ref	XML Element	Base Standard Usage	Base Rationale	Profile Usage	Reference & Remarks
1	<S11:Header>	mr	Necessary to include<wsse:Security>.	mr	
1.1	<wsse:Security>	mr	Necessary to support digital signatures and encryption.	mr	
1.1.1	S11:actor	c[6]	Section 5 of wsse indicates an S11:actor may be omitted but in only one header and that no two headers can have the same actor or role.	c	If message security processing logic is implemented as handlers within the SOAP stack, they should be considered as part of the destination rather than a separate intermediary. Therefore, the *actor* attribute may be omitted to indicate the header is targeted to that recipient.
1.1.2	S11:mustUnderstand	o	Section 5 of wsse implies this attribute may be absent.	mr	

[6] mr if no other <wsse:Security> has S11:actor present, else m.

Ref	XML Element	Base Standard Usage	Base Rationale	Profile Usage	Reference & Remarks
1.1.3	Tokens	o	Section 6 of wsse implies these elements may be absent.	mr	See Table A.4
1.1.4	<ds:Signature>	c[7]	Required only if performing digital signature.	m	Some but not all environments will require signed/encrypted/signed—for example, signing a secret document, encrypting the document (thereby rendering it unclassified), and then signing again.
1.1.5	<xenc:EncryptedData>	c[8]	Required only if performing encryption operations.	c	See Table A.7
2	<S11:Body>	mr	Required to support detached signatures.	mr	

[7] mr for digital signature operations, else o.

[8] mr for encryption operations, else o.

Table A.3 S12 Envelope

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
1	<S12:Header>	mr	Necessary to include<wsse:Security>.	i	
1.1	<wsse:Security>	mr	Necessary to support digital signatures and encryption.	i	
1.1.1	S12:role	c[9]	Section 5 of wsse indicates an S12:role may be omitted but in only one header and that no two headers can have the same actor or role.	i	This attribute carries the same semantics as S11:actor
1.1.2	S12:mustUnderstand	o	Section 5 of wsse implies this attribute may be absent.	i	
1.1.3	Tokens	o	Section 6 of wsse implies these elements may be absent.	i	See Table A.4

[9] mr if no other <wsse:Security> has S12:role present, else m.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
1.1.4	<ds:Signature>	c[10]	Required only if performing digital signature.	i	Some but not all environments will require signed/encrypted/signed—for example, signing a secret document, encrypting the document (thereby rendering it unclassified), and then signing again.
1.1.5	<xenc:EncryptedData>	c[11]	Required only if performing encryption operations.	i	See Table A.7
2	<S12:Body>	mr	Required to support detached signatures.	i	

[10] mr for digital signature operations, else o.

[11] mr for encryption operations, else o.

Table A.4 Tokens

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
1	<wsse:UsernameToken>	o	Section 6.2.1 of wsse specifies that this is an optional element.	x	
1.2	<wsu:Id>	o	String label for token required for references. Section 3.2 of wsu indicates the Id is used.	i	Because the top level element of UsernameToken is not supported, all child elements remain out of scope.
1.3	<wsse:Username>	mr	Section 6.2.1 of wsse indicates this element is required.	i	
1.4	<wsse:Password>	o	Section 3.1 of wsu indicates this element is optional.	i	
1.4.1	Type	o	Section 3.1 of wsu indicates this attribute is optional.	i	
1.4.1.1	PasswordText	m	Section 3.1 of wsu specifies this value as the default.	i	
1.4.1.2	PasswordDigest	o	Section 3.1 of wsu indicates this value is optional.	i	

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
1.5	<wsse:Nonce>	o	Section 3.1 of wsu indicates this element is optional.	i	
1.5.1	EncodingType	o	Section 3.1 of wsu indicates this attribute is optional.	i	See Table A.8/2
1.6	<wsu:Created>	o	Section 3.1 of wsu indicates this element is optional.	i	
2	<wsse:BinarySecurityToken>	o	Section 6.3.1 of wsse indicates this element is optional.	mr	Must address support for multiple PKI
2.1	<wsu:Id>	o	Section 6.3.2 of wsse indicates this attribute is optional.	mr	Note must be a shorthand Xpointer Reference.
2.2	ValueType	m	Section 6.3.2 of wsse indicates this attribute is used to indicate the value space and is recommended.	mr	See Table A.8/1
2.3	EncodingType	m	Section 6.3.2 of wsse indicates this attribute is used to indicate the encoding format.	mr	See Table A.8/2

Table A.5 ds:Signature

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
1	Id	o	Section 4.1 of dsig indicates this attribute is optional.	o	
2	<ds:SignedInfo>	mr	Section 4.1 of dsig indicates this element is mandatory.	mr	
2.1	Id	o	Section 4.3 of dsig indicates this attribute is optional.	o	
2.2	<ds:CanonicalizationMethod>	mr	Section 4.3.1 of dsig indicates this element is required.	mr	See Table A.8/3
2.3	<ds:SignatureMethod>	mr	Section 4.3.2 of dsig indicates this element is required.	mr	
2.3.1	Algorithm	mr	Section 6.1 of dsig indicates this attribute is required.	mr	See Table A.9 for a listing of RFC-defined signature types
2.3.2	<ds:HMACOutputLength>	c[12]	Section 6.3.1 of dsig indicates this element is optional.	x	

[12] mr if Algorithm is a MAC Algorithm, else o.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
2.4	<ds:Reference>	mr	Section 4.3 of dsig indicates this element is mandatory.	mr	Indicate which parts of the message are included in the signature calculation
2.4.1	Id	o	Section 4.3.3 of dsig indicates this attribute is optional.	o	
2.4.2	URI	o	Section 4.3.3 of dsig indicates this attribute is optional.	mr	
2.4.3	Type	o	Section 4.3.3 of dsig indicates this attribute is optional.	o	
2.4.4	<ds:Transforms>	o	Section 4.3.3.4 of dsig indicates this element is optional.	mr	See Table A.8/4
2.5	<ds:DigestMethod>	mr	Section 4.3.3.5 of dsig indicates this element is required.	mr	
2.5.1	Algorithm	mr	Section 6.1 of dsig indicates this element is required.	mr	See Table A.9 for a list of RFC-defined digest algorithm identifiers
2.6	<ds:DigestValue>	mr	Section 4.3.3.6 of dsig indicates this element is required.	mr	
3	<ds:SignatureValue>	mr	Section 4.1 of dsig indicates this element is mandatory.	mr	

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
3.1	Id	o	Section 4.2 of dsig indicates this attribute is optional.	o	
4	<ds:KeyInfo>	o	Section 4.4 of dsig indicates this element is optional.	mr	Note references to X509 PKCs will be used instead.
4.1	Id	o	Section 4.4 of dsig indicates this attribute is optional.	o	
4.2	<ds:KeyName>	o	Section 4.4 of dsig indicates this element is optional.	x	May occur one or more times.
4.3	<ds:KeyValue>	m	Section 4.4 of dsig indicates this element must be supported.	x	All child elements should also be prohibited because the top level element is prohibited
4.3.1	<ds:DSAKeyValue>	m	Section 4.4.2 and 6.1 of dsig indicates this is the required algorithm.	x	
4.3.1.1	P	c[13]	Section 4.4.2.1 of dsig indicates this element is optional but P and Q must either both appear or both be absent.	x	

[13] m if P and Q absent, else mr.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
4.3.1.2	Q	c[13]	Section 4.4.2.1 of dsig indicates this element is optional but P and Q must either both appear or both be absent.	x	
4.3.1.3	G	m	Section 4.4.2.1 of dsig indicates this element is optional.	x	
4.3.1.4	Y	mr	Section 4.4.2.1 of dsig indicates this element must be present.	x	
4.3.1.5	J	mr	Section 4.4.2.1 of dsig indicates this element must be present if P and Q are present.	x	
4.3.1.6	seed	c[14]	Section 4.4.2.1 of dsig indicates this element is optional but seed and pgenCounter must either both appear or both be absent.	x	

[14] m if seed and pgenCounter absent, else mr.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
4.3.1.7	pgenCounter	c[14]	Section 4.4.2.1 of dsig indicates this element is optional but seed and pgenCounter must either both appear or both be absent.	x	
4.3.2	<ds:RSAKeyValue>	mr	Section 4.4.2 and 6.1 of dsig indicates this is a recommended algorithm.	x	May occur one or more times.
4.3.2.1	Modulus	mr	Section 4.4.2.2 of dsig indicates this element is required.	x	
4.3.2.2	Exponent	mr	Section 4.4.2.2 of dsig indicates this element is required.	x	
4.4	<ds:RetrievalMethod>	m	Section 4.4 of dsig indicates this element should be supported.	x	May occur one or more times.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
4.4.1	URI	mr	Section 4.4.3 of dsig indicates URI is required from RetrievalMethod.	x	
4.4.2	Type	c[15]	Section 3.2 of RFC4051 indicates this attribute is required if one of the “raw” types is used.	x	See Table A.8/5
4.4.3	<ds:Transforms>	o	Section 4.4.3 of dsig indicates this element is optional.	x	See Table A.8/4
4.5	<ds:X509Data>	o	Section 4.4 of dsig indicates this element is optional.	x	The X509 PKCs will be carried in BinarySecurityToken.
4.5.1	<ds:X509IssuerSerial>	o	Section 4.4.4 of dsig indicates this element is optional.	x	Note one must be supported.
4.5.1.1	<ds:X509IssuerName>	mr	Section 4.4.4 of dsig indicates this element is required.	x	Note one must be supported. Should be compliant with RFC2253.
4.5.1.2	<ds:X509SerialNumber>	mr	Section 4.4.4 of dsig indicates this element is required.	x	

[15] mr if “raw” type, else o.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
4.5.2	<ds:X509SubjectName>	o	Section 4.4.4 of dsig indicates this element is optional.	x	Note one must be supported. Should be compliant with RFC2253.
4.5.3	<ds:X509SKI>	o	Section 4.4.4 of dsig indicates this element is optional.	x	Note one must be supported. Note: X509 refers to this as X509SubjectKeyIdentifier.
4.5.4	<ds:X509Certificate>	o	Section 4.4.4 of dsig indicates this element is optional.	x	Note one must be supported.
4.5.5	<ds:X509CRL>	o	Section 4.4.4 of dsig indicates this element is optional.	x	Note one must be supported.
4.6	<ds:PGPData>	o	Section 4.4 of dsig indicates this element is optional.	x	May occur one or more times.
4.6.1	<ds:PGPKeyId>	c[16]	Section 4.4 of dsig indicates this element is required if PGPKeyPacket is absent.	x	

[16] mr if PGPKeyPacket absent, else m.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
4.6.2	<ds:PGPKeyPacket>	c[17]	Section 4.4 of dsig indicates this element is required if PGPKeyId is absent.	x	
4.7	<ds:SPKIData>	o	Section 4.4 of dsig indicates this element is optional.	x	May occur one or more times.
4.7.1	<ds:SPKISexp>	mr	Section 4.4.6 of dsig indicates this element is mandatory.	x	
4.8	<ds:MgmtData>	o	Section 4.4 of dsig indicates this element is optional. Use of this element is not recommended.	x	May occur one or more times.
4.9	<wsse:SecurityTokenReference>	m	Section 7.1 of wsse recommends this element be used here.	mr	See Table A.6
5	<ds:Object>	o	Section 4.5 of dsig indicates this element is optional.	x	May occur one or more times.
5.1	Id	o	Section 4.5 of dsig indicates this attribute is optional.	x	Not supported because the parent element is not supported

[17] mr if PGPKeyId absent, else o.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
5.2	MimeType	o	Section 4.5 of dsig indicates this attribute is optional.	x	
5.3	Encoding	o	Section 4.5 of dsig indicates this attribute is optional.	x	
6	<ds:Manifest>	o	Section 5.1 of dsig indicates this element is optional.	x	
6.1	Id	o	Section 5.1 of dsig indicates this element is optional.	x	
6.2	<ds:References>	mr	Section 5.1 of dsig indicates this element is required.	x	
7	<ds:SignatureProperties>	o	Section 5.2 of dsig indicates this element is optional.	o	Information about the signature may be helpful for RBAC
7.1	Id	o	Section 5.2 of dsig indicates this element is optional.	o	
7.2	<ds:SignatureProperty>	mr	Section 5.2 of dsig indicates this element is required.	mr	The parent element itself is optional. Therefore, this is mandatory only when the parent is also present.
7.2.1	Id	o	Section 5.2 of dsig indicates this element is optional.	o	

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
7.2.2	Target	mr	Section 5.2 of dsig indicates this element is required.	mr	

Table A.6 wsse:SecurityTokenReference

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
1	<wsu:Id>	m	Section 7.1 of wsse does not indicate that it is optional.	m	
2	Usage	o	Section 7.1 of wsse indicates this attribute is optional.	o	
3	<wsse:Reference>	c[18]	Section 7.2 of wsse indicates this element is the mechanism for referencing using URIs.	mr	
3.1	URI	o	Section 7.2 of wsse indicates this attribute is optional.	mr	

[18] mr if URI, else o.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
3.2	ValueType	o	Section 7.2 of wsse indicates this attribute is optional. It does recommend use when tokens are local.	mr	See Table A.8/1
4	<wsse:KeyIdentifier>	c[19]	Section 7.3 of wsse indicates this element shall be the mechanism for referencing using key identifiers.	o	
4.1	<wsu:Id>	o	Section 7.3 of wsse indicates this attribute is optional.	o	
4.2	ValueType	o	Section 7.3 of wsse indicates this attribute is optional.	o	See Table A.8/6
4.3	EncodingType	o	Section 7.3 of wsse indicates this attribute is optional.	o	See Table A.8/2
5	<wsse:Embedded>	c[20]	Section 7.4 of wsse indicates this element is used for local references.	o	

[19] mr if key identifier, else o.

[20] mr if embedded, else o.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
5.1	<wsu:Id>	o	Section 7.4 of wsse indicates this attribute is optional.	o	
6	<ds:KeyInfo>	m	Section 7.1 of wsse indicates this element can be used to carry information but recommends BinarySecurityToken be used instead.	m	See Table A.5/4

Table A.7 EncryptedData

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
1	ID	o	Section 3.1 of xenc indicates this attribute is optional.	i	
2	Type	o	Section 3.1 of xenc indicates this attribute is optional.	i	
3	MimeType	o	Section 3.1 of xenc indicates this attribute is optional.	i	
4	Encoding	o	Section 3.1 of xenc indicates this attribute is optional.	i	

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
5	EncryptionMethod	o	Section 3.1 of xenc indicates this element is optional.	i	
5.1	Algorithm	mr	Section 3.2 of xenc indicates this element is required.	i	See Table A.9
5.2	xenc:KeySize	m	Section 3.2 of xenc indicates this element is mandatory.	i	
5.3	OAEPparams	c[21]	Section 3.2 of xenc indicates this element is required.	i	
5.4	<ds:Digest>	c[21]	Section 5.4.2 of xenc indicates this element is required for RSA-OAEP.	i	
6	<xenc:CipherData>	mr	Section 3.1 of xenc indicates this element is required.	i	
6.1	CipherValue	c[22]	Section 3.3 of xenc indicates this element is required if xenc:CipherReference is absent.	i	

[21] m if Algorithm=RSA-OAEP, else o.

[22] mr if xenc:CipherReference absent, else o.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
6.2	<xenc:CipherReference>	c[23]	Section 3.3 of xenc indicates this element is required if CipherValue is absent.	i	
6.2.1	URI	mr	Section 3.3.1 of xenc indicates this extension is required.	i	
6.2.2	Transforms	o	Section 3.3.1 of xenc indicates this element is optional.	i	
6.2.2.1	Transform	mr	Section 3.3.1 of xenc indicates this element is optional.	i	See Table A.8/4
7	EncryptionProperties	o	Section 3.1 of xenc indicates this element is optional.	i	
7.1	Id	o	Section 3.7 of xenc indicates this attribute is optional.	i	
7.2	EncryptionProperty	mr	Section 3.7 of xenc indicates this element is required.	i	
7.2.1	Id	o	Section 3.7 of xenc indicates this attribute is optional.	i	

[23] mr if CipherValue absent, else o.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
7.2.2	Target	o	Section 3.7 of xenc indicates this attribute is optional.	i	

Table A.8 Data Values

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
1	TokenType				
1.1	X509Token	o	No requirement in wsse, dsig, or X509 for this token type.	mr	
1.1.1	X509v1	o	Section 3.1 of X509 indicates this token type is optional.	x	
1.1.2	X509v3	o	Section 3.1 of X509 indicates this token type is optional.	m	
1.1.2	X509PKIPathv1	o	Section 3.1 of X509 indicates this token type is optional.	x	
1.1.3	PKCS7	o	Section 3.1 of X509 indicates this token type is optional.	i	

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
1.2	UsernameToken	o	No requirement in wsse, dsig, or pass for this token type.	x	
1.3	SAMLToken	o	No requirement in wsse or dsig for this token type.	c	
1.4	XrMLToken (Rights Expressive Language)	o	No requirement in wsse or dsig for this token type.	i	
1.5	XCBF (Common Biometric Format)	o	No requirement in wsse or dsig for this token type.	i	
1.6	Kerberosv5TGT	o	No requirement in wsse or dsig for this token type.	i	
1.7	Kerberosv5TST	o	No requirement in wsse or dsig for this token type.	i	
1.8	LTPAToken (Lightweight Third-party Auth.)	o	No requirement in wsse or dsig for this token type.	i	
2	EncodingType				
2.1	Base64Binary	m	Section 6.3.2 of wsse specifies this as the default.	mr	
2.2	Hex	o	No requirement for Hex encoding.	o	
3	CanonicalizationMethod				

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
3.1	Inclusive Canonical XML	mr	Section 6.5.1 of dsig indicates this form is required.	x	
3.2	Inclusive Canonical XML with Comments	m	Section 6.5.1 of dsig indicates this form is recommended	x	
3.3	Exclusive Canonical XML	mr	Section 8.1 of wsse indicates this form is strongly recommended.	m	With comments is preferred (See Section 4.5).
3.4	Exclusive Canonical XML with Comments	o	Section 5.1 of xenc indicates this form is optional	mr	Preferred (See Section 4.5).
4	Transforms				
4.1	Inclusive Canonical XML	mr	Section 6.6 of dsig recommends this transform be supported.	x	
4.2	Inclusive Canonical XML with Comments	m	Section 6.6 of dsig recommends this transform be supported.	x	
4.3	Exclusive Canonical XML	mr	Section 8.1 of wsse indicates this transform is required.	m	With comments is preferred (See Section 4.5).
4.4	Exclusive Canonical XML with Comments	m	Section 5.1 of xenc indicates this form be supported.	mr	Preferred (See Section 4.5).

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
4.5	Base64	m	Section 6.6 of dsig recommends this transform be supported.	i	
4.6	XPath Filtering	m	Section 6.6 of dsig recommends this transform be supported.	i	
4.7	Enveloped Signature Transform	m	Section 6.6 of dsig recommends this transform be supported.	i	
4.8	XSLT Transform	m	Section 6.6 of dsig recommends this transform be supported.	x	
4.9	SOAP Message Normalization	o	Section 8.1 of wsse indicates this form may be supported. Note this is a non-normative section.	i	
5	Retrieval Method Types				
5.1	DSAKeyValue	c[24]	Necessary only if DSAKeyValue is retrieved.	i	

[24] mr if ds:KeyInfo is DSAKeyValue, else o.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
5.2	RSAKeyValue	c[25]	Necessary only if RSAKeyValue is retrieved.	i	
5.3	X509Data	c[26]	Necessary only if X509Data is retrieved.	i	
5.4	PGPData	c[27]	Necessary only if PGPData is retrieved.	i	
5.5	SPKIData	c[28]	Necessary only if SPKIData is retrieved.	i	
5.6	MgmtData	c[29]	Necessary only if MgmtData is retrieved.	x	
5.7	PKCS7signedData	c[30]	Only necessary if PKCS7signedData is retrieved.	i	
5.8	rawX509Certificate	c[31]	Section 3.2 of RFC4051 indicates this attribute value is required if the retrieved type is “raw”.	i	

[25] mr if ds:KeyInfo is RSAKeyValue, else o.

[26] mr if ds:KeyInfo is X509Data, else o.

[27] mr if ds:KeyInfo is PGPData, else o.

[28] mr if ds:KeyInfo is SPKIData, else o.

[29] mr if ds:KeyInfo is MgmtData, else o.

[30] mr if ds:KeyInfo is PKCS7signedData, else o.

[31] mr if ds:KeyInfo is rawX509Certificate, else o.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
5.9	rawX509CRL	c[32]	Section 3.2 of RFC4051 indicates this attribute value is required if the retrieved type is “raw”.	i	
5.10	rawPGPKeyPacket	c[33]	Section 3.2 of RFC4051 indicates this attribute value is required if the retrieved type is “raw”.	i	
5.11	rawSPKISexp	c[34]	Section 3.2 of RFC4051 indicates this attribute value is required if the retrieved type is “raw”.	i	
5.12	rawPKCS7signedData	c[35]	Section 3.2 of RFC4051 indicates this attribute value is required if the retrieved type is “raw”.	i	
5.13	encryptedKey	o	Section 3.5.2 of xenc indicates this type is optional.	c	

[32] mr if ds:KeyInfo is rawX509XCRL, else o.

[33] mr if ds:KeyInfo is rawPGPPacket, else o.

[34] mr if ds:KeyInfo is rawSPKISexp, else o.

[35] mr if ds:KeyInfo is rawPKCS7signedData, else o.

Ref	XML Element	Base Standard Usage	Base Standard Rationale	Profile Usage	Reference & Remarks
6	KeyIdentifier ValueType				

Table A.9 Algorithms

IETF RFC 4051, “Additional XML Security Uniform Resource Identifiers (URIs)” provides a convenient reference list of URIs and descriptions for algorithms in which there is substantial interest, but which cannot or have not been included in the XML Digital Signature specification. All of the algorithms in RFC 4051 are legal values according to XMLDSig syntax because they have been assigned a unique URI identifier by the IETF. Each of the algorithms within RFC 4051 that are not assigned a unique identifier directly in XMLDSig are assigned the following namespace: http://www.w3.org/2001/04/xmldsig-more#. See Section 4.5.1 for a description of the required algorithms. Any other algorithm is considered out of scope and SHOULD not be supported.