WS-SecurityPolicy Examples

Committee Draft 01

4 June 2009

Specification URIs:
This Version:
http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cd-01.doc (Authoritative)
http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cd-01.pdf
http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cd-01.html

Previous Version:
N/A

Latest Version:
http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples.doc
http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples.pdf
http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples.html

Technical Committee:
OASIS Web Services Secure Exchange TC

Chair(s):
Kelvin Lawrence, IBM
Chris Kaler, Microsoft

Editor(s):
Rich Levinson, Oracle Corporation
Tony Gullotta, SOA Software Inc.
Symon Chang, BEA Systems Inc.
Martin Raepple, SAP AG

Related work:
N/A

Declared XML Namespace(s):
N/A

Abstract:

This document contains examples of how to set up WS-SecurityPolicy [WSSP] policies for a variety of common token types that are described in WS-Security 1.0 [WSS10] and WS-Security 1.1 [WSS11] token profiles [WSSTC]. Particular attention is focused on the different "security bindings" (defined in [WSSP]) within the example policies. Actual messages that have been documented in WS-Security TC [WSSTC]and other WS-Security-based Interops [WSSINTEROPS, WSSXINTEROPS, OTHERINTEROPS] that conform to some of the example policies are referenced when appropriate.
The purpose of this document is to give examples of how policies may be defined for several existing use cases that have been part of the WS-Security Interops that have been conducted (see References section for Interop documents [INTEROPS]). In addition, some example use cases have been included which show some variations from the WS-Security Interop use cases in order to demonstrate how different options and security bindings impact the structure of the policies.

Status:

This document was last revised or approved by the WS-SX TC on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at http://www.oasis-open.org/committees/ws-sx/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/ws-sx/ipr.php.

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/ws-sx/.

Notices

Table of Contents

1 Introduction

This document describes several WS-SecurityPolicy [WS-SECURITYPOLICY] examples. An example typically consists of the security aspects of a high-level Web Service use-case with several variable components. Many of the examples are based on existing use cases that have been conducted during WS-Security Interops [WSS-INTEROPS]. In those examples a reference is included to identify the specific use case in the specific interop document that is being demonstrated.

In the policy examples below, the "wsp" prefix refers to the elements defined in the WS-Policy namespace:

xmlns:wsp="http://www.w3.org/ns/ws-policy/"

the "sp" prefix refers to elements defined in the WS-SecurityPolicy (WS-SP) namespace:

xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"

the "t" prefix refers to elements defined in the WS-Trust namespace:

xmlns:t=" http://docs.oasis-open.org/ws-sx/ws-trust/200512"

Where uses cases are based on existing scenarios, those scenarios are referenced at the beginning of the use case section. The explicit documents describing the scenarios are identified by links in [Section 3.2].

1.1 Terminology and Concepts

This section (1.1.*) describes the logical "actors" that participate in the examples. In addition, there is a discussion on general concepts that describes how the logical actors typically relate to the physical message exchanges that take place.

This section also provides a security reference model designed to provide context for the examples in terms of a conceptual framework within which the actors interact, which is intended to help readers understand the trust relationships implicit in the message exchanges shown in the examples.

In these examples there are always three important conceptual entities to recognize that exist on the initiating side of a transaction, where the transaction is being requested by sending an electronic message that contains the details of the what is being requested and by whom (the "entities" become "actors" as the discussion moves from the conceptual to the specific). These three entities are:

- The entity **requesting** the transaction (for example, if the transaction is about an application for a home mortgage loan, then the entity requesting the transaction is the prospective homeowner who will be liable to make the payments on the loan if it is approved).
- The entity **approving** the transaction to be requested. This entity is generally known as an "identity provider", or an "authority", and the purpose of this approving entity is to guarantee to a recipient entity that the requesting entity is making a legitimate request (continuing the above example, the authorizing entity in this case would be the organization that helps the prospective homeowner properly fill out the application, possibly a loan officer at the bank, saying that the loan application is approved for further processing).
- The entity **initiating** the actual electonic transaction message (in the above example, this entity is simply the technical software used to securely transmit the mortgage application request to a **recipient** entity that will handle the processing of the mortgage application).

WS-SecurityPolicy is primarily concerned with the technical software used between the initiating entity and the recipient entity, whom are respectively officially referred to as the Initiator and the Recipient in the WS-SecurityPolicy 1.2 specification (WS-SP) (see section 1.4 of that document).

Therefore, the purpose of this section is to give a larger real world context to understanding the examples and how to relate the technical details of WS-SecurityPolicy to the actual logical actors involved in the transactions governed by the technology.

The reason for providing this context is to help interested readers understand that while the technology may be securing the integrity and confidentiality of the messages, there are additional questions about transactions such as who is liable for any commitments resulting from the transaction and how those commitments are technically bound within the message exchanges.

The purpose here is not to provide unequivocable answers to all questions regarding liability of transactions, but to give a sense of how the structuring of a request message ties the participating entities to the transaction. Depending on how the WS-SecurityPolicy technology is used, these “ties” can be relatively stronger or weaker. Depending on the nature of the transactions supported by a particular Web Application, the system managers of the Web Services Application being protected using WS-SecurityPolicy techniques, may be interested in a conceptual framework for understanding which WS-SP techniques are more or less appropriate for their needs.

These introductory sections are intended to provide this type of conceptual framework for understanding how the examples in this document may be interpreted in the above context of the three entities on the initiating side of the transaction. A complementary model is also provided for the recipient side of the transaction, but since the recipient is generally concerned with validating the request, which is primarily a back office function, less emphasis is focused on the options in that environment except as they might relate back to the initiator side of the transaction.

1.1.1 Actors

The following diagram shows the actors and the relationships that may be typically involved in a network security scenario:

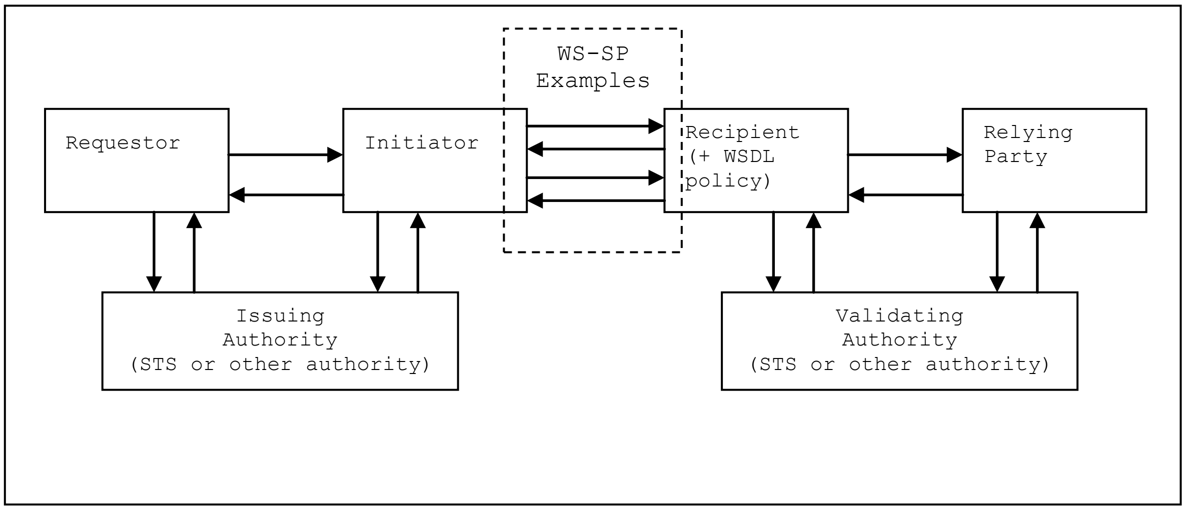
WS-SP Examples
Requestor
Initiator
Recipient (+ WSDL policy)
Relying Party
Issuing Authority (STS or other authority)
Validating Authority (STS or other authority)

Figure 1.

The diagram is intended to show the possible interactions that may occur between actors in any given scenario, although, in general, depending on the policy specified by the recipient, only a subset of the possible interactions will actually occur in a given scenario. Note that the Issuing and Validating Authorities, may, in general be either a WS-Trust Security Token Service (STS) or other authority.

First, a brief narrative will describe the diagram, then the actors will be defined in more detail.

In a typical example interaction, a Requestor wants to issue a Web Service request to a Web Service that is hosted by the “Recipient” web site on behalf of a RelyingParty, who is actually the business entity responsible for making the service available and with whom any business arrangements with the Requestor are made. One may think of this as an end to end interaction between a Requestor and a

RelyingParty, with technical resources being provided by the Initiator on the Requestor side and the Recipient on the RelyingParty side.

Technically, what occurs is that the Requestor hands the request to the Initiator, which in turn issues a query to the Recipient and is returned the WSDL [WSDL] describing the Web Service, which generally includes WS-SP policy Assertions [WS-POLICY, WS-POLICYATTACHMENT] that describe the security policies supported by the Recipient for this Web Service (Note: it is not necessary that the information in the Assertions be obtained this way. It may also be stored locally based on out of band agreements between the Requestor and RelyingParty). This interaction is shown by the upper pair of arrows between the Initiator and the Recipient.

Upon receipt of the WS-SP policy assertions, the Initiator then interacts with the Requestor and the Issuing Authority, as needed in order to meet the requirements specified by the WS-SP. Generally, what is required here is that credentials and tokens are obtained from the Requestor and Issuing Authority and assembled as required in a new WS-Security request message that is to be issued to the Recipient.

(For example, if a UsernameToken is required by the Recipient, one possibility is that the Initiator will query the Requestor for Username and Password and create the appropriate token to include in the Request.

Other possibilities exist as well, but WS-SP only governs what the final message must contain. How it gets constructed and how the pieces are assembled are specific to the system environment that is being used.

In general, the examples in this document will explain the interactions that might occur to meet the policy requirements, but the actual sequences and specifics will be determined by the setup of the systems involved.)

Finally, after assembling the necessary tokens, the Initiator (or Requestor/Initiator) may sign and encrypt the message as required by the WS-SP policy and send it to the Recipient.
Similar to the Requestor side, on the Recipient side, the details of how the Recipient processes the message and uses a Validating Authority to validate the tokens and what basis the RelyingParty uses to accept or reject the request is system specific. However, in a general sense, the 3 actors identified on the Recipient side will be involved to accept and process a request.

(For example, the Recipient may decrypt an encrypted UsernameToken containing a clear text password and username and pass it to the Validating Authority for validation and authentication, then the Recipient may pass the Request to the RelyingParty, which may in turn issue a request for authorization to the Validating Authority.

These details are beyond the scope of WS-SP and the examples in this document, however, knowing that this is the context in which WS-SP operates can be useful to understanding the motivation and usefulness of different examples.)
The following list is a reference to identify the logical actors in Figure 1. (In general, these actors may physically be implemented such that more than one actor is included in the physical entity, such as a Security Token Service (STS) [WS-TRUST] that implements both an IssuingAuthority and a ValidatingAuthority. Similarly, in a scenario where a user is at a security-enabled work station, the work station may combine a Requestor and Initiator in a single physical entity.)

- **Requestor:** the person or entity requesting the service and who will be supplying credentials issued by the IssuingAuthority and will be validated by a ValidatingAuthority. The Requestor is the logical entity that supplies the credentials that ultimately get passed to the Recipient that will be trusted by the RelyingParty if they are validated by the ValidatingAuthority. (Note: the logistics of supplying the trusted credential is distinct from the logistics of packaging up the credentials within a WS-Security header in a SOAP message and transmitting that message to the Recipient. The latter logistics are covered by the Initiator, described below. The Requestor and Initiator may be combined into a single physical entity and this is a common occurrence, however they do not have to be and it is also a common occurrence that they are separate physical entities. The latter case is typified by the Requestor being a user at a browser that may be promped for credentials by the Initiator on a server using HTTP to challenge the Requestor for a username and password.)

- **IssuingAuthority:** a Security Token Service (STS), which is an organization or entity that typically issues authentication credentials for Requestors. Examples include X509 Certificate Authority, Saml Token Authority, Kerberos Token Authority. (For user passwords, the IssuingAuthority may be thought of as the entity the user contacts for password services, such as changing password, setting reminder phrases, etc.)
- **Initiator:** the system or entity that sends the message(s) to the Recipient requesting use of the service on behalf of the Requestor. Typically, the Initiator will first contact the Recipient on behalf of the Requestor and obtain the WS-SP policy and determine what tokens from what kind of IssuingAuthority (X509, SAML, Kerberos, etc) that the Requestor will require to access the service and possibly assist the Requestor to obtain those credentials. In addition, based on the WS-SP policy, the Initiator determines how to format the WS-Security headers of the messages being sent and how to use the security binding required by the policy.
- **Recipient:** the system or entity or organization that provides a web service for use and is the supplier of the WS-SP policy that is contained in each example and is the recipient of messages sent by Initiators. The Recipient manages all the processing of the request message. It may rely on the services of a ValidatingAuthority to validate the credentials contained in the message. When the Recipient has completed the WS-SP directed processing of the message it will generally be delivered to the RelyingParty which continues processing of the message based on the assurances that the Recipient has established by verifying the message is in compliance with the WS-SP policies that are attached to the service description.
- **ValidatingAuthority:** the organization or entity that typically validates Requestor credentials for Relying Parties, and, in general, maintains those credentials in an ongoing reliable manner. Typically, the Recipient will request the ValidatingAuthority to validate the credentials before delivering the Request to the RelyingParty for further processing.
- **RelyingParty:** the organization or entity that relies on the security tokens contained in the messages sent by the Initiator as a basis for providing the service. For this document, the RelyingParty may simply be considered to be the entity that continues processing the message after the Recipient has completed all the processing required by the WS-SP policies attached to the service description.

Of these actors, the Requestor and Initiator can generally being regarded as “client-side” or “requestor-side” actors. The Recipient and RelyingParty (or combined “**RelyingParty/Recipient**”) can be regarded as “server-side” actors.

> Note 1: In addition to the above labelling of the actors, there is also the notion of “**Sender**”. Generally, the “Sender” may be thought of as a Requestor/Initiator that is independently collecting information from a user (who could be modeled as a separate benign actor to the left of the Requestor in the figure 1.1 from whom the Requestor gathers information that would be included in the message) and is submitting requests on behalf of that user. Generally, the trust of the Recipient is on the Sender, and it is up to the Sender to do whatever is necessary for the Sender to trust the user. Examples of this configuration will be described in the SAML Sender Vouches sections later in this document.

> Note 2: The person or entity actually making the request is the "Requestor", however, there are 2 common use cases: 1. the Requestor is a user at a browser and the request is intercepted by a web service that can do WS-Security and this web service is the "Initiator" which actually handles the message protection and passes the Requestor’s credentials (typically collected via http(s) challenge by the Initiator to the Requestor for username/password) to the Recipient. 2. the Requestor is at a web-service client enabled workstation, where the Requestor person making the request is also in charge of the web service client that is initiating the request, in which case the combined entity may be referred to as a "**Requestor/Initator**".

1.1.2 Concepts

Physical message exchanges are between the Initiator and Recipient. For the purposes of this document the Initiator and Recipient may be considered to be the physical systems that exchange the messages. The Initiator and Recipient use the keys that are involved in the WS-SP security binding that protects the messages.

As described in the previous section, the Requestor is the logical entity that gathers the credentials to be used in the request and the Initiator is the logical entity that inserts the credentials into the request message and does all the rest of the message construction in accordance with the WS-SP policy. The Requestor may generally be thought of as being either a separate physical entity from the Initiator, or as part of a combined physical entity with the Initiator. An example of the latter combined case would be a user at a client workstation equipped with signing and encryption capabilities, where the combined entity may be referred to as a “Requestor/Initiator”.

Similarly, the IssuingAuthority should generally be thought of as a separate physical entity from the Initiator. However, in some cases, such as SAML sender-vouches, the IssuingAuthority and the Initiator may be the same entity.

In some other cases, such as the case where user passwords are involved, the ValidatingAuthority system entity may also comprise the Recipient and the Relying Party, since passwords are typically checked locally for validity.

The focus of WS-SP is the notion that policy assertions are attached to the Initiator and/or Recipient, however, the concepts in those policies generally require understanding specifically the relation of the parties involved (i.e. Requestor/Initiator, RelyingParty/Recipient). This is because the Requestor in general does not know in advance what policies each Web Service provider requires and it is necessary for practical purposes to have a front end Initiator resolve the policy and coordinate whatever actions are required to exchange the Requestor tokens for the tokens required by the service. This token exchange may be done using WS-Trust [WS-TRUST] to prepare the necessary requests and process responses from an STS IssuingAuthority. Examples of these WS-Trust token exchanges may be found in [WSSX-INTEROP].

Typically both the Requestor/Initiator and Recipient/RelyingParty will have relations with the IssuingAuthority/ValidatingAuthority and often establish contact with those Authorities using WS-Trust for the purpose of obtaining and validating tokens used in their Web Service interactions. The policies for using the Authority may also be represented using WS-SP, but they are typically no different from the policies shown in the examples in this document. The policies in this document may be used for any kind of request, whether it be a request to a service provider for general content or operations or a request to an Authority for authentication tokens.

In each example the relations between these actors and how the request message is prepared will be described, because, in general, the policy requirements will imply these relationships. Generally, each of the 3 client side actors, the Requestor, the Initiator, and the IssuingAuthority will contribute to the preparation of the request message, which is the primary focus of this document. For validation of the message, the Recipient, the RelyingParty, and the ValidatingAuthority are generally involved, but for this document the focus is simply that the Recipient provides the WS-SP policy that dictates the preparation of the request message.

1.1.2.1 X509 Certificates

The specifics of whom is trusted for X509 certificates depends on the specific organization’s PKI (Public Key Infrastructure) setup. For this document, we shall assume the Subject of the X509 certificate identifies the actor, which may be the IssuingAuthority, or the Initiator, or the Requestor, depending on the example. We shall also assume that the issuer of the X509 certificates is a general Certificate Authority not directly involved in any authorization of the web service transactions, but is relied on for the validity of the X509 certificate in a manner out of scope of the scenarios covered. In addition, this document does not explicitly address the case of X509 certificates issued by the IssuingAuthority actor. Such use cases are generally implicitly covered if one assumes that such relations are automatically covered by the specifics of the organization PKI setups.

However, the IssuingAuthority may issue tokens, such as SAML holder-of-key that contain X509 certificates. In these cases, the basis of trust is that the X509 Certificate of the IssuingAuthority was used to protect the X509 certificate of the Requestor which is contained in the signed SAML holder-of-key token. I.e. any “chaining” of tokens is done by referencing those tokens within signed XML structures and not by issuing actual X509 certificates

1.2 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in **[RFC2119]**.

1.3 Normative References

[RFC2119]	S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.
[WSS10-SOAPMSG]	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
[WSS11-SOAPMSG]	http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
[WSS10-USERNAME]	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf
[WSS11-USERNAME]	http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf
[WSS10-X509-PROFILE]	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf
[WSS11-X509-PROFILE]	http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf
[WSS10-SAML11-PROFILE]	http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf
[WSS11-SAML1120-PROFILE]	http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-218-spec-os-SAMLTokenProfile.pdf
[WSS11-LIBERTY-SAML20-PROFILE]	http://www.projectliberty.org/liberty/content/download/894/6258/file/liberty-idwsf-security-mechanisms-saml-profile-v2.0.pdf
[WSS-TC-ALL-TOKEN-PROFILES]	http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
[WS-SECURITYPOLICY]	http://www.oasis-open.org/specs/index.php#wssecpolv1.2
[WS-SECURECONVERSATION]	http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.pdf
[WS-TRUST]	http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf
[WS-POLICY]	http://www.w3.org/TR/ws-policy
[WS-POLICYATTACHMENT]	http://www.w3.org/TR/ws-policy-attach
[WSDL]	http://www.w3.org/TR/wsdl
[SSL]	http://www.ietf.org/rfc/rfc2246.txt
[SAML11-CORE]	http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf
[SAML20-CORE]	http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf
[XML-DSIG]	http://www.w3.org/TR/xmldsig-core/
[XML-ENCR]	http://www.w3.org/TR/xmlenc-core/

1.4 Non-Normative References

WSS10-INTEROP-01: http://www.oasis-open.org/committees/download.php/11374/wss-interop1-draft-06-merged-changes.pdf

WSS10-INTEROP-02: http://www.oasis-open.org/committees/download.php/11375/wss-interop2-draft-06-merged.doc

WSS11-INTEROP-01: http://www.oasis-open.org/committees/download.php/12997/wss-11-interop-draft-01.doc

WSS10-KERBEROS-INTEROP: http://www.oasis-open.org/committees/download.php/10991/wss-kerberos-interop.doc

WSS10-SAML11-INTEROP: http://www.oasis-open.org/committees/download.php/7702/wss-saml-interop1-draft-12.doc

WSS11-SAML1120-INTEROP: http://www.oasis-open.org/committees/download.php/16556/wss-saml2-interop-draft-v4.doc

WSSX-PRE-INTEROP: http://www.oasis-open.org/committees/download.php/16357/Trust_SecureConversation_Interop.2004-10.doc

WSSX-WSTR-WSSC-INTEROP: http://www.oasis-open.org/committees/download.php/20954/ws-sx-interop-ed-10.doc

WS-SECURE-INTEROP: http://www.oasis-open.org/committees/document.php?document_id=28803&wg_abbrev=ws-sx

WSI-SCM-SAMPLEAPPL: http://www.ws-i.org/SampleApplications/SupplyChainManagement/2006-04/SCMSecurityArchitectureWGD5.00.doc (login required)

1.5 Specifications

1.6 Interops and Sample Messages

2 Scenarios

2.1 UsernameToken

UsernameToken authentication scenarios that use simple username password token for authentication. There are several sub-cases.

2.1.1 UsernameToken – no security binding

In this model a UsernameToken is placed within a WS-Security header in the SOAP Header [WSS10-USERNAME, WSS11-USERNAME]. No other security measure is used.

Because no security binding is used, there is no explicit distinction between the Requestor, who is identified in the UsernameToken and the Initiator, who physically sends the message. They may be one and the same or distinct parties. The lack of a security binding indicates that any direct URL access method (ex. HTTP) may be used to access the service.

2.1.1.1 UsernameToken with plain text password

This scenario is based on the first WS-Security Interop Scenarios Document [WSS10-INTEROP-01 Scenario 1 – section 3.4.4]

(http://www.oasis-open.org/committees/download.php/11374/wss-interop1-draft-06-merged-changes.pdf).

This policy says that Requestor/Initiator must send a password in a UsernameToken in a WS-Security header to the Recipient (who as the Authority will validate the password). The password is required because that is the default requirement for the Web Services Security Username Token Profile 1.x [WSS10-USERNAME, WSS11-USERNAME].

This setup is only recommended where confidentiality of the password is not an issue, such as a pre-production test scenario with dummy passwords, which might be used to establish that the Initiator can read the policy and prepare the message correctly, and that connectivity and login to the service can be performed.

```
(P001)   <wsp:Policy>
(P002)     <sp:SupportingTokens>
(P003)       <wsp:Policy>
(P004)         <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient"
(P005)       </wsp:Policy>
(P006)     </sp:SupportingTokens>
(P007)   </wsp:Policy>
```

An example of a message that conforms to the above stated policy is as follows.

```
(M001)    <?xml version="1.0" encoding="utf-8" ?>
(M002)    <soap:Envelope xmlns:soap="...">
(M003)      <soap:Header>
(M004)        <wsse:Security soap:mustUnderstand="1" xmlns:wsse="...">
(M005)          <wsse:UsernameToken>
(M006)            <wsse:Username>Chris</wsse:Username>
(M007)            <wsse:Password Type="http://docs.oasis-
   open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
   1.0#PasswordText">sirhC</wsse:Password>
(M008)            <wsse:Nonce EncodingType="...#Base64Binary"
(M009)               >pN...=</wsse:Nonce>
```

```
(M010)            <wsu:Created>2007-03-28T18:42:03Z</wsu:Created>
(M011)          </wsse:UsernameToken>
(M012)        </wsse:Security>
(M013)      </soap:Header>
(M014)      <soap:Body>
(M015)        <Ping xmlns="http://xmlsoap.org/Ping">
(M016)          <text>EchoString</text>
(M017)        </Ping>
(M018)      </soap:Body>
(M019)    </soap:Envelope>
```

The UsernameToken element starting on line (M005) satisfies the UsernameToken assertion on line (P004). By default, a Password element is included in the UsernameToken on line (M007) holding a plain text password. Lines (M008-M010) contain an optional Nonce element and Created timestamp, which, while optional, are recommended to improve security of requests against replay and other attacks [WSS10-USERNAME]. All WS-Security compliant implementations should support the UsernameToken with cleartext password with or without the Nonce and Created elements.

2.1.1.2 UsernameToken without password

This policy is the same as 2.1.1.1 except no password is to be placed in the UsernameToken. There are no credentials to further establish the identity of the Requestor and no security binding that the Initiator is required to use. This is a possible production scenario where all the service provider wants is a UsernameToken to associate with the request. There is no explicit Authority implied in this scenario, except possibly that the username extracted from the UsernameToken would be evaluated by a server-side “Authority” that maintained a list of valid username values.

```
(P001)    <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
(P002)      <sp:SupportingTokens>
(P003)        <wsp:Policy>
(P004)          <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P005)            <wsp:Policy>
(P006)              <sp:NoPassword/>
(P007)            </wsp:Policy>
(P008)          </sp:UsernameToken>
(P009)        </wsp:Policy>
(P010)      </sp:SupportingTokens>
(P011)    </wsp:Policy>
```

Lines (P002) – (P010) contain the SupportingToken assertion which includes a UsernameToken indicating that a UsernameToken must be included in the security header.

Line (P006) requires that the wsse:Password element must not be present in the UsernameToken.

An example of an input message that conforms to the above stated policy is as follows:

```
(M001)    <?xml version="1.0" encoding="utf-8" ?>
(M002)    <soap:Envelope xmlns:soap="...">
(M003)      <soap:Header>
(M004)        <wsse:Security soap:mustUnderstand="1"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
(M005)          <wsse:UsernameToken>
(M006)            <wsse:Username>Chris</wsse:Username>
(M007)          </wsse:UsernameToken>
(M008)        </wsse:Security>
(M009)      </soap:Header>
(M010)      <soap:Body>
```

```
(M011)       <Ping xmlns="http://xmlsoap.org/Ping">
(M012)         <text>EchoString</text>
(M013)       </Ping>
(M014)     </soap:Body>
(M015)   </soap:Envelope>
```

Lines (M005) – (M007) hold the unsecured UsernameToken which only contains the name of user (M006), but no password.

2.1.1.3 UsernameToken with timestamp, nonce and password hash

This scenario is similar to 2.1.1.1, except it is more secure, because the Requestor password is protected by combining it with a nonce and timestamp, and then hashing the combination. Therefore, this may be considered as a potential production scenario where passwords may be safely used. It may be assumed that the password must be validated by a server-side ValidatingAuthority and so must meet whatever requirements the specific Authority has established.

```
(P001)   <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
(P002)     <sp:SupportingTokens>
(P003)       <wsp:Policy>
(P004)         <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P005)           <wsp:Policy>
(P006)             <sp:HashPassword/>
(P007)           </wsp:Policy>
(P008)         </sp:UsernameToken>
(P009)       </wsp:Policy>
(P010)     </sp:SupportingTokens>
(P011)   </wsp:Policy>
```

An example of a message that conforms to the above stated policy is as follows.

```
(M001)   <?xml version="1.0" encoding="utf-8" ?>
(M002)   <soap:Envelope xmlns:soap="..." xmlns:wsu=”...”>
(M003)     <soap:Header>
(M004)       <wsse:Security soap:mustUnderstand=“1” xmlns:wsse="...">
(M005)         <wsse:UsernameToken
                    wsu:Id="uuid-7cee5976-0111-e9c1-e34b-af1e85fa3866">
(M006)           <wsse:Username>Chris</wsse:Username>
(M007)           <wsse:Password  Type="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-usernametoken-profile-
1.0#PasswordDigest"
                    >weYI3nXd8LjMNVksCKFV8t3rgHh3Rw==</wsse:Password>
(M008)           <wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
(M009)           <wsu:Created>2007-05-01T01:15:30Z</wsu:Created>
(M010)         </wsse:UsernameToken>
(M011)       </wsse:Security>
(M012)     </soap:Header>
(M013)     <soap:Body wsu:Id="uuid-7cee4264-0111-e0cb-8329-af1e85fa3866">
(M014)       <Ping xmlns="http://xmlsoap.org/Ping">
(M015)         <text>EchoString</text>
(M016)       </Ping>
(M017)     </soap:Body>
(M018)   </soap:Envelope>
```

This message is very similar to the one in section 2.1.1.1. A UsernameToken starts on line (M005) to satisfy the UserNameToken assertion. However, in this example the Password element on line (M007) is of type PasswordDigest to satisfy the HashPassword assertion on line (P006). The Nonce (M008) and Created timestamp (M009) are also included as dictated by the HashPassword assertion. The Nonce and timestamp values are included in the password digest on line (M007).

2.1.2 Use of SSL Transport Binding

Both server authentication and mutual (client AND server) authentication SSL [SSL] are supported via use of the sp:TransportBinding policy assertion. (For mutual authentication, a RequireClientCertificate assertion may be inserted within the HttpsToken assertion. The ClientCertificate may be regarded as a credential token for authentication of the Initiator, which in the absence of any additional token requirements would generally imply the Initiator is also the Requestor. The Authority would be the issuer of the client certificate.)

```
<wsp:Policy xmlns:wsp="..." xmlns:sp="...">
  <sp:TransportBinding>
    <wsp:Policy>
      <sp:TransportToken>
        <wsp:Policy>
          <sp:HttpsToken />
        </wsp:Policy>
      </sp:TransportToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          <sp:Basic256 />
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
    </wsp:Policy>
  </sp:TransportBinding>
</wsp:Policy>
```

2.1.2.1 UsernameToken as supporting token

Additional tokens can be included as supporting tokens. Each of the UsernameTokens described in section 2.1.1 may be used in this scenario and any clear text information or password will be protected by SSL. So, for example, including a user name token over server authentication SSL we have:

```
(P001) <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
(P002)   <sp:TransportBinding>
(P003)     <wsp:Policy>
(P004)       <sp:TransportToken>
(P005)         <wsp:Policy>
(P006)           <sp:HttpsToken/>
(P007)         </wsp:Policy>
(P008)       </sp:TransportToken>
(P009)       <sp:AlgorithmSuite>
(P010)         <wsp:Policy>
(P011)           <sp:Basic256/>
(P012)         </wsp:Policy>
(P013)       </sp:AlgorithmSuite>
(P014)       <sp:Layout>
(P015)         <wsp:Policy>
(P016)           <sp:Strict/>
(P017)         </wsp:Policy>
(P018)       </sp:Layout>
(P019)       <sp:IncludeTimestamp/>
```

```
(P020)     </wsp:Policy>
(P021)   </sp:TransportBinding>
(P022)   <sp:SupportingTokens>
(P023)     <wsp:Policy>
(P024)       <sp:UsernameToken/>
(P025)     </wsp:Policy>
(P026)   </sp:SupportingTokens>
(P027) </wsp:Policy>
```

Lines (P002) – (P021) contain the TransportBinding assertion which indicates that the message must be protected by a secure transport protocol like SSL or TLS.

Lines (P004) – (P008) hold the TransportToken assertion, indicating that the transport is secured by means of an HTTPS Transport Token, requiring to perform cryptographic operations based on the transport token using the Basic256 algorithm suite (P011).

In addition, the Layout assertion in lines (P014) – (P018) require that the order of the elements in the SOAP message security header must conform to rules defined by WSSECURITYPOLICY that follow the general principle of 'declare before use'.

Line (P019) indicates that the wsu:Timestamp element must be present in the SOAP message security header.

Lines (P022) – (P026) Lines contain the SupportingToken assertion which includes a UsernameToken indicating that a UsernameToken must be included in the security header.

An example of an input message prior to the transport encryption that conforms to the above stated policy is as follows:

```
(M001) <?xml version="1.0" encoding="utf-8" ?>
(M002) <soap:Envelope xmlns:soap="..." xmlns:wsu="...">
(M003)   <soap:Header>
(M004)     <wsse:Security soap:mustUnderstand="1"
              xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
              wss-wssecurity-secext-1.0.xsd">
(M005)       <wsu:Timestamp wsu:Id="uuid-8066364f-0111-f371-47bf-ba986d2d7dc4">
(M006)         <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
(M007)       </wsu:Timestamp>
(M008)       <wsse:UsernameToken
                 wsu:Id="uuid-8066368d-0111-e744-f37b-ba986d2d7dc4">
(M009)         <wsse:Username>Chris</wsse:Username>
(M010)         <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-1.0#PasswordText">sirhC</wsse:Password>
(M011)       </wsse:UsernameToken>
(M012)     </wsse:Security>
(M013)   </soap:Header>
(M014)   <soap:Body wsu:Id="uuid-8066363f-0111-ffdc-de48-ba986d2d7dc4">
(M015)     <Ping xmlns="http://xmlsoap.org/Ping">
(M016)       <text>EchoString</text>
(M017)     </Ping>
(M018)   </soap:Body>
(M019) </soap:Envelope>
```

Lines (M005) – (M007) hold Timestamp element according to the IncludeTimestamp assertion.

Lines (M008) – (M011) hold the UsernameToken which contains the name (M009) and password (M010) of the user sending this message.

2.1.3 (WSS 1.0) UsernameToken with Mutual X.509v3 Authentication, Sign, Encrypt

This scenario is based on WS-I SCM Security Architecture Technical requirements for securing the SCM Sample Application, March 2006 [WSI-SCM-SAMPLEAPPL – GetCatalogRequest, SubmitOrderRequest].

This use case corresponds to the situation where both parties have X.509v3 certificates (and public-private key pairs). The Initiator includes a user name token that may stand for the Requestor on-behalf-of which the Initiator is acting. The UsernameToken is included as a SupportingToken; this is also encrypted. The Authority for this request is generally the Subject of the Initiator's trusted X.509 Certificate.

We model this by using the asymmetric security binding [WSSP] with a UsernameToken SupportingToken.

The message level policies in this section and subsequent sections cover a different scope of the web service definition than the security binding level policy and so appear as separate policies and are attached at WSDL Message Policy Subject. These are shown below as input and output policies. Thus, we need a set of coordinated policies one with endpoint subject and two with message subjects to achieve this use case.

The policy is as follows:

```
(P001) <wsp:Policy wsu:Id="wss10_up_cert_policy" >
(P002)   <wsp:ExactlyOne>
(P003)     <wsp:All>
(P004)       <sp:AsymmetricBinding>
(P005)         <wsp:Policy>
(P006)           <sp:InitiatorToken>
(P007)             <wsp:Policy>
(P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-
open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P009)                 <wsp:Policy>
(P010)                   <sp:WssX509V3Token10/>
(P011)                 </wsp:Policy>
(P012)               </sp:X509Token>
(P013)             </wsp:Policy>
(P014)           </sp:InitiatorToken>
(P015)           <sp:RecipientToken>
(P016)             <wsp:Policy>
(P017)               <sp:X509Token sp:IncludeToken="http://docs.oasis-
open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
(P018)                 <wsp:Policy>
(P019)                   <sp:WssX509V3Token10/>
(P020)                 </wsp:Policy>
(P021)               </sp:X509Token>
(P022)             </wsp:Policy>
(P023)           </sp:RecipientToken>
(P024)           <sp:AlgorithmSuite>
(P025)             <wsp:Policy>
(P026)               <sp:Basic256/>
(P027)             </wsp:Policy>
(P028)           </sp:AlgorithmSuite>
(P029)           <sp:Layout>
(P030)             <wsp:Policy>
(P031)               <sp:Strict/>
(P032)             </wsp:Policy>
(P033)           </sp:Layout>
(P034)           <sp:IncludeTimestamp/>
(P035)           <sp:OnlySignEntireHeadersAndBody/>
(P036)         </wsp:Policy>
(P037)       </sp:AsymmetricBinding>
(P038)       <sp:Wss10>
(P039)         <wsp:Policy>
```

```
(P040)          <sp:MustSupportRefKeyIdentifier/>
(P041)        </wsp:Policy>
(P042)      </sp:Wss10>
(P043)      <sp:SignedEncryptedSupportingTokens>
(P044)        <wsp:Policy>
(P045)          <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P046)            <wsp:Policy>
(P047)              <sp:WssUsernameToken10/>
(P048)            </wsp:Policy>
(P049)          </sp:UsernameToken>
(P050)        </wsp:Policy>
(P051)      </sp:SignedEncryptedSupportingTokens>
(P052)    </wsp:All>
(P053)  </wsp:ExactlyOne>
</wsp:Policy>

(P054) <wsp:Policy wsu:Id="WSS10UsernameForCertificates_input_policy">
(P055)   <wsp:ExactlyOne>
(P056)     <wsp:All>
(P057)       <sp:SignedParts>
(P058)         <sp:Body/>
(P059)       </sp:SignedParts>
(P060)       <sp:EncryptedParts>
(P061)         <sp:Body/>
(P062)       </sp:EncryptedParts>
(P063)     </wsp:All>
(P064)   </wsp:ExactlyOne>
(P065) </wsp:Policy>

(P066) <wsp:Policy wsu:Id="WSS10UsernameForCertificate_output_policy">
(P067)   <wsp:ExactlyOne>
(P068)     <wsp:All>
(P069)       <sp:SignedParts>
(P070)         <sp:Body/>
(P071)       </sp:SignedParts>
(P072)       <sp:EncryptedParts>
(P073)         <sp:Body/>
(P074)       </sp:EncryptedParts>
(P075)     </wsp:All>
(P076)   </wsp:ExactlyOne>
(P077) </wsp:Policy>
```

Lines (P004) – (P037) contain the AsymmetricBinding assertion which indicates that the initiator's token must be used for the message signature and the recipient's token must be used for message encryption.

Lines (P006) – (P014) contain the InitiatorToken assertion. Within that assertion lines (P008) – (P012) indicate that the initiator token must be an X.509 token that must be included with all messages sent to the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as described in the WS-Security 1.0 X.509 Token Profile.

Lines (P015) – (P023) contain the RecipientToken assertion. Within that assertion lines (P017) – (P021) dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509 Token Profile, however as stated on line (P017) it must not be included in any message. Instead, according to the MustSupportKeyRefIdentitier assertion on line (P040) a KeyIdentifier must be used to identify the token in any messages where the token is used.

Line (P034) requires the inclusion of a timestamp.

Lines (P043) – (P051) contain a SignedEncryptedSupportingTokens assertion which identifies the inclusion of an additional token which must be included in the message signature and encrypted. Lines (P045) – (P049) indicate that the supporting token must be a UsernameToken and must be included in all messages to the recipient. Line (P047) dictates the UsernameToken must conform to the WS-Security 1.0 UsernameToken Profile.

Lines (P055) – (P066) contain a policy that is attached to the input message. Lines (P058) – (P060) require that the body of the input message must be signed. Lines (P061) – (P063) require the body of the input message must be encrypted.

Lines (P067) – (P078) contain a policy that is attached to the output message. Lines (P070) – (P072) require that the body of the output message must be signed. Lines (P073) – (P075) require the body of the output message must be encrypted.

An example of an input message that conforms to the above stated policy is as follows.

```
(M001) <?xml version="1.0" encoding="utf-8" ?>
(M002) <soap:Envelope xmlns:soap="..." xmlns:xenc="..." xmlns:ds="...">
(M003)   <soap:Header>
(M004)     <wsse:Security soap:mustUnderstand="1" xmlns:wsse="..." xmlns:wsu="...">
(M005)       <xenc:ReferenceList>
(M006)         <xenc:DataReference URI="#encUT"/>
(M007)         <xenc:DataReference URI="#encBody"/>
(M008)       </xenc:ReferenceList>
(M009)       <wsu:Timestamp wsu:Id="T0">
(M010)         <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
(M011)       </wsu:Timestamp>
(M012)       <wsse:BinarySecurityToken wsu:Id="binaryToken" ValueType="…#X509v3"
EncodingType="…#Base64Binary">
(M013)         MIIEZzCCA9CgAwIBAgIQEmtJZc0…
(M014)       </wsse:BinarySecurityToken>
(M015)       <xenc:EncryptedData wsu:Id="encUT">
(M016)         <ds:KeyInfo>
(M017)           <wsse:SecurityTokenReference>
(M018)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
ValueType="...#X509SubjectKeyIdentifier">
(M019)               MIGfMa0GCSq…
(M020)             </wsse:KeyIdentifier>
(M021)           </wsse:SecurityTokenReference>
(M022)         </ds:KeyInfo>
(M023)         <xenc:CipherData>
(M024)           <xenc:CipherValue>...</xenc:CipherValue>
(M025)         </xenc:CipherData>
(M026)       </xenc:EncryptedData>
(M027)       <ds:Signature>
(M028)         <ds:SignedInfo>…
(M029)           <ds:Reference URI="#T0">…</ds:Reference>
(M030)           <ds:Reference URI="#usernameToken">…</ds:Reference>
(M031)           <ds:Reference URI="#body">…</ds:Reference>
(M032)         </ds:SignedInfo>
(M033)         <ds:SignatureValue>HFLP…</ds:SignatureValue>
(M034)         <ds:KeyInfo>
(M035)           <wsse:SecurityTokenReference>
(M036)             <wsse:Reference URI="#binaryToken"/>
(M037)           </wsse:SecurityTokenReference>
(M038)         </ds:KeyInfo>
(M039)       </ds:Signature>
(M040)     </wsse:Security>
(M041)   </soap:Header>
(M042)   <soap:Body wsu:Id="body">
(M043)     <xenc:EncryptedData wsu:Id="encBody">
(M044)       <ds:KeyInfo>
(M045)         <wsse:SecurityTokenReference>
(M046)           <wsse:KeyIdentifier EncodingType="...#Base64Binary"
ValueType="...#X509SubjectKeyIdentifier">
(M047)             MIGfMa0GCSq…
(M048)           </wsse:KeyIdentifier>
(M049)         </wsse:SecurityTokenReference>
(M050)       </ds:KeyInfo>
(M051)       <xenc:CipherData>
(M052)         <xenc:CipherValue>...</xenc:CipherValue>
```

```
(M053)       </xenc:CipherData>
(M054)     </xenc:EncryptedData>
(M055)   </soap:Body>
(M056) </soap:Envelope>
```

Line (M006) is an encryption data reference that references the encrypted UsernameToken on lines (M015) – (M024) which was required to be included by the SignedEncryptedSupportingTokens assertion. Lines (M018) – (M020) hold a KeyIdentifier of the recipient's token used to encrypt the UsernameToken as required by the AsymmetricBinding assertion. Because the RecipientToken assertion disallowed the token from being inserted into the message, a KeyIdentifier is used instead of a reference to an included token.

Line (M007) is an encryption data reference that references the encrypted body of the message on lines (M043) – (M054). The encryption was required by the EncryptedParts assertion of the input message policy. It also uses the recipient token as identified by the KeyIdentifier.

Lines (M009) – (M011) contain a timestamp for the message as required by the IncludeTimestamp assertion.

Lines (M012) – (M014) contain the BinarySecurityToken holding the X.509v3 certificate of the initiator as required by the InitiatorToken assertion.

Lines (M027) – (M039) contain the message signature.

Line (M029) indicates the message timestamp is included in the signature as required by the IncludeTimestamp assertion definition.

Line (M030) indicates the supporting UsernameToken is included in the signature as required by the SignedEncryptedSupportingTokens assertion. Because the token was encrypted its content prior to encryption is included below to better illustrate the reference.

```
(M057)  <wsse:UsernameToken wsu:Id="usernameToken">
(M058)    <wsse:Username>Chris</wsse:Username>
(M059)    <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">sirhC</wsse:Password>
(M060)  </wsse:UsernameToken>
```

Line (M031) indicates the message body is included in the signature as required by the SignedParts assertion of the input message policy.

Note that the initiator's BinarySecurityToken is not included in the message signature as it was not required by policy.

Line (M036) references the initiator's BinarySecurityToken included in the message for identifying the key used for signing as dictated by the AsymmetricBinding assertion.

2.1.3.1 (WSS 1.0) Encrypted UsernameToken with X.509v3

This scenario is based on the first WS-Security Interop Scenarios Document [WSS10-INTEROP-01 Scenario 2 – section 4.4.4]

(http://www.oasis-open.org/committees/download.php/11374/wss-interop1-draft-06-merged-changes.pdf).

This policy says that Requestor/Initiator must send a password in an encrypted UsernameToken in a WS-Security header to the Recipient (who as the Authority will validate the password). The password is required because that is the default requirement for the Web Services Security Username Token Profile 1.x [WSS10-USERNAME, WSS11-USERNAME].

This setup is only recommended when the sender cannot provide the "message signature" and it is RECOMMENDED that the receiver employs some security mechanisms external to the message to prevent the spoofing attacks.

The policy is as follows:

```
(P001) <wsp:Policy wsu:Id="wss10_encrypted_unt_policy" >
(P002)   <sp:AsymmetricBinding>
(P003)     <wsp:Policy>
(P004)       <sp:InitiatorEncryptionToken>
```

```
(P005)         <wsp:Policy>
(P006)           <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
(P007)             <wsp:Policy>
(P008)               <sp:WssX509V3Token10/>
(P009)             </wsp:Policy>
(P010)           </sp:X509Token>
(P011)         </wsp:Policy>
(P012)       </sp:InitiatorEncryptionToken>
(P013)       <sp:RecipientSignatureToken>
(P014)         <wsp:Policy>
(P015)           <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
(P016)             <wsp:Policy>
(P017)               <sp:WssX509V3Token10/>
(P018)             </wsp:Policy>
(P019)           </sp:X509Token>
(P020)         </wsp:Policy>
(P021)       </sp:RecipientSignatureToken>
(P022)       <sp:AlgorithmSuite>
(P023)         <wsp:Policy>
(P024)           <sp:Basic256/>
(P025)         </wsp:Policy>
(P026)       </sp:AlgorithmSuite>
(P027)       <sp:Layout>
(P028)         <wsp:Policy>
(P029)           <sp:Lax/>
(P030)         </wsp:Policy>
(P031)       </sp:Layout>
(P032)       <sp:IncludeTimestamp/>
(P033)       <sp:OnlySignEntireHeadersAndBody/>
(P034)     </wsp:Policy>
(P035)  </sp:AsymmetricBinding>
(P036)  <sp:Wss10>
(P037)    <wsp:Policy>
(P038)      <sp:MustSupportRefKeyIdentifier/>
(P039)      <sp:MustSupportRefIssuerSerial/>
(P040)    </wsp:Policy>
(P041)   </sp:Wss10>
(P042)   <sp:EncryptedSupportingTokens>
(P043)     <wsp:Policy>
(P044)       <sp:UsernameToken sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P045)         <wsp:Policy>
(P046)           <sp:HashPassword/>
(P047)           <sp:WssUsernameToken10/>
(P048)         </wsp:Policy>
(P049)       </sp:UsernameToken>
(P050)     </wsp:Policy>
(P051)   </sp:EncryptedSupportingTokens>
(P052) </wsp:Policy>

(P053) <wsp:Policy wsu:Id="WSS10UsernameForCertificates_input_policy">
(P054)   <wsp:ExactlyOne>
(P055)     <wsp:All>
(P056)       <sp:EncryptedParts>
(P057)         <sp:Body/>
(P058)       </sp:EncryptedParts>
(P059)     </wsp:All>
(P060)   </wsp:ExactlyOne>
(P061) </wsp:Policy>

(P062) <wsp:Policy wsu:Id="WSS10UsernameForCertificate_output_policy">
(P063)   <wsp:ExactlyOne>
```

```
(P064)     <wsp:All>
(P065)       <sp:SignedParts>
(P066)         <sp:Body/>
(P067)       </sp:SignedParts>
(P068)     </wsp:All>
(P069)   </wsp:ExactlyOne>
(P070) </wsp:Policy>
```

Lines (P002) – (P035) contain the AsymmetricBinding assertion which indicates that the recipient's token must be used for both message signature and encryption.

Lines (P004) – (P012) contain the InitiatorEncryptionToken assertion. Within that assertion lines (P006) – (P010) indicate that the initiator token must be an X.509 token. Line (P008) dictates the X.509 token must be an X.509v3 security token as described in the WS-Security 1.0 X.509 Token Profile, however as stated on line (P006) it must not be included in any message.

Lines (P013) – (P021) contain the RecipientSignatureToken assertion. Within that assertion lines (P015) – (P019) dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509 Token Profile for message signature, however as stated on line (P017) it must not be included in any message.

Line (P032) requires the inclusion of a timestamp.

Line (P035) OnlySignEntireHeadersAndBody assertion will only apply to the response message, as there is no signature token defined for the Initiator.

Lines (P036) – (P041) contain some WS-Security 1.0 related interoperability requirements, specifically support for key identifier, and issuer serial number.

Lines (P042) – (P051) contain an EncryptedSupportingTokens assertion which identifies the inclusion of an additional token which must be included in the message and encrypted. Lines (P044) – (P049) indicate that the supporting token must be a UsernameToken and must be included in all messages to the recipient. Line (P046) dictates the UsernameToken must be hash into digest format, instead of clear text password. Line (P047) dictates the UsernameToken must conform to the WS-Security 1.0 UsernameToken Profile.

Lines (P053) – (P061) contain a policy that is attached to the input message. Lines (P056) – (P058) require the body of the input message must be encrypted.

Lines (P062) – (P070) contain a policy that is attached to the output message. Lines (P065) – (P068) require the body of the output message must be signed.

An example of a request message that conforms to the above stated policy is as follows.

```
(M001) <?xml version="1.0" encoding="utf-8" ?>
(M002) <soapenv:Envelope  xmlns:soapenv="..."   xmlns:xenc="..." . . . >
(M003)   <soapenv:Header>
(M004)     <wsse:Security xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="..."
soapenv:mustUnderstand="1"  >
(M005)       <xenc:EncryptedKey>
(M006)         <xenc:EncryptionMethod Algorithm=". . .#rsa-oaep-mgf1p">
(M007)           <ds:DigestMethod Algorithm=". . .#sha1" />
(M008)         </xenc:EncryptionMethod>
(M009)         <ds:KeyInfo>
(M010)           <wsse:SecurityTokenReference >
(M011)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
(M012)               ValueType="...#X509SubjectKeyIdentifier">CuJ. .
.=</wsse:KeyIdentifier>
(M013)           </wsse:SecurityTokenReference>
(M014)         </ds:KeyInfo>
(M015)         <xenc:CipherData>
(M016)           <xenc:CipherValue>dbj...=</xenc:CipherValue>
(M017)         </xenc:CipherData>
(M018)         <xenc:ReferenceList>
(M019)           <xenc:DataReference URI="#encBody"/>
(M020)           <xenc:DataReference URI="#encUnt"/>
```

```
(M021)         </xenc:ReferenceList>
(M022)       </xenc:EncryptedKey>
(M023)       <xenc:EncryptedData Id="encUnt" Type="...#Element"
MimeType="text/xml" Encoding="UTF-8" >
(M024)         <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
(M025)         <xenc:CipherData>
(M026)           <xenc:CipherValue>KZf...=</xenc:CipherValue>
(M027)         </xenc:CipherData>
(M028)       </xenc:EncryptedData>
(M029)       <wsu:Timestamp >
(M030)         <wsu:Created>2007-03-28T18:42:03Z</wsu:Created>
(M031)       </wsu:Timestamp>
(M032)     </wsse:Security>
(M033)   </soapenv:Header>
(M034)   <soapenv:Body >
(M035)     <xenc:EncryptedData Id="encBody" Type="...#Content" MimeType="text/xml"
Encoding="UTF-8" >
(M036)       <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
(M037)       <xenc:CipherData>
(M038)         <xenc:CipherValue>a/9...B</xenc:CipherValue>
(M039)       </xenc:CipherData>
(M040)     </xenc:EncryptedData>
(M041)   </soapenv:Body>
(M042) </soapenv:Envelope>
```

Line (M020) is an encryption data reference that references the encrypted UsernameToken on lines (M023) – (M028) which was required to be included by the EncryptedSupportingTokens assertion. Lines (M009) – (M014) hold a KeyIdentifier of the recipient's token used to encrypt the UsernameToken as required by the AsymmetricBinding assertion. Because the InitiatorEncryptionAssertion disallowed the token from being inserted into the message, a KeyIdentifier is used instead of a reference to an included token.

Line (M019) is an encryption data reference that references the encrypted body of the message on lines (M035) – (M040). The encryption was required by the EncryptedParts assertion of the input message policy. It also uses the recipient token as identified by the KeyIdentifier.

Lines (M029) – (M031) contain a timestamp for the message as required by the IncludeTimestamp assertion.

Because the username token was encrypted its content prior to encryption is included below to better illustrate the reference.

```
(M043)  <wsse:UsernameToken wsu:Id="usernameToken">
(M044)    <wsse:Username>Chris</wsse:Username>
(M045)    <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordDigest">oY...=</wsse:Password>
(M046)    <wsse:Nonce EncodingType="...#Base64Binary">pN...=</wsse:Nonce>
(M047)    <wsu:Created>2007-03-28T18:42:03Z</wsu:Created>
(M048)  </wsse:UsernameToken>
```

Line (M046) contains the Nonce element and Line (M047) contains a timestamp. It is recommended that these two elements should also be included in the PasswordText case for better security [WSS10-USERNAME].

2.1.4 (WSS 1.1), User Name with Certificates, Sign, Encrypt

This scenario is based on the "Examples of Secure Web Service Message Exchange Document" [WS-SECURE-INTEROP].

The use case here is the following: the Initiator generates a symmetric key; the symmetric key is encrypted using the Recipient's certificate and placed in an encrypted key element. The UsernameToken identifying the Requestor and message body are signed using the symmetric key. The body and

UsernameToken are also encrypted. The Authority for this request is generally the Subject of the Initiator’s X509 certificate.

We can use the symmetric security binding [WSSP] with X509token as the protection token to illustrate this case. If derived keys are to be used, then the derived keys property of X509Token should be set.

The policy is as follows:

```
(P001)   <wsp:Policy wsu:Id="WSS11UsernameWithCertificates_policy">
(P002)     <wsp:ExactlyOne>
(P003)       <wsp:All>
(P004)         <sp:SymmetricBinding>
(P005)           <wsp:Policy>
(P006)             <sp:ProtectionToken>
(P007)               <wsp:Policy>
(P008)                 <sp:X509Token sp:IncludeToken="http://docs.oasis-
open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never”>
(P009)                   <wsp:Policy>
(P010)                     <sp:RequireThumbprintReference/>
(P011)                     <sp:WssX509V3Token11/>
(P012)                   </wsp:Policy>
(P013)                 </sp:X509Token>
(P014)               </wsp:Policy>
(P015)             </sp:ProtectionToken>
(P016)             <sp:AlgorithmSuite>
(P017)               <wsp:Policy>
(P018)                 <sp:Basic256/>
(P019)               </wsp:Policy>
(P020)             </sp:AlgorithmSuite>
(P021)             <sp:Layout>
(P022)               <wsp:Policy>
(P023)                 <sp:Strict/>
(P024)               </wsp:Policy>
(P025)             </sp:Layout>
(P026)             <sp:IncludeTimestamp/>
(P027)             <sp:OnlySignEntireHeadersAndBody/>
(P028)           </wsp:Policy>
(P029)         </sp:SymmetricBinding>
(P030)         <sp:SignedEncryptedSupportingTokens>
(P031)           <wsp:Policy>
(P032)             <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P033)               <wsp:Policy>
(P034)                 <sp:WssUsernameToken11/>
(P035)               </wsp:Policy>
(P036)             </sp:UsernameToken>
(P037)           </wsp:Policy>
(P038)         </sp:SignedEncryptedSupportingTokens>
(P039)         <sp:Wss11>
(P040)           <wsp:Policy>
(P041)             <sp:MustSupportRefKeyIdentifier/>
(P042)             <sp:MustSupportRefIssuerSerial/>
(P043)             <sp:MustSupportRefThumbprint/>
(P044)             <sp:MustSupportRefEncryptedKey/>
(P045)           </wsp:Policy>
(P046)         </sp:Wss11>
(P047)       </wsp:All>
(P048)     </wsp:ExactlyOne>
(P049)   </wsp:Policy>

(P050)   <wsp:Policy wsu:Id="UsernameForCertificates_input_policy">
(P051)     <wsp:ExactlyOne>
(P052)       <wsp:All>
(P053)         <sp:SignedParts>
(P054)           <sp:Body/>
(P055)         </sp:SignedParts>
```

```
(P056)         <sp:EncryptedParts>
(P057)           <sp:Body/>
(P058)         </sp:EncryptedParts>
(P059)       </wsp:All>
(P060)     </wsp:ExactlyOne>
(P061)   </wsp:Policy>

(P062)   <wsp:Policy wsu:Id="UsernameForCertificate_output_policy">
(P063)     <wsp:ExactlyOne>
(P064)       <wsp:All>
(P065)         <sp:SignedParts>
(P066)           <sp:Body/>
(P067)         </sp:SignedParts>
(P068)         <sp:EncryptedParts>
(P069)           <sp:Body/>
(P070)         </sp:EncryptedParts>
(P071)       </wsp:All>
(P072)     </wsp:ExactlyOne>
(P073)   </wsp:Policy>
```

Lines (P004) – (P297) contain a SymmetricBinding assertion which indicates the use of one token to both sign and encrypt a message.

Lines (P006) – (P015) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P012) indicate that the protection token must be an X.509 token that must never be included in any messages in the message exchange. Line (P010) dictates the X.509 token must be an X.509v3 security token as described in the WS-Security 1.1 X.509 Token Profile. Line (P011) dicates a thumbprint reference must be used to identify the token in any message.

Line (P026) requires the inclusion of a timestamp.

Lines (P030) – (P038) contain a SignedEncryptedSupportingTokens assertion which identifies the inclusion of an additional token which must be included in the message signature and encrypted. Lines (P032) – (P036) indicate that the supporting token must be a UsernameToken and must be included in all messages to the recipient. Line (P034) dictates the UsernameToken must conform to the WS-Security 1.1 UsernameToken Profile.

Lines (P040) – (P046) contain some WS-Security 1.1 related interoperability requirements, specifically support for key identifier, issuer serial number, thumbprint, and encrypted key references.

Lines (P050) – (P061) contain a policy that is attached to the input message. Lines (P053) – (P055) require that the body of the input message must be signed. Lines (P056) – (P058) require the body of the input message must be encrypted.

Lines (P062) – (P073) contain a policy that is attached to the output message. Lines (P065) – (P067) require that the body of the output message must be signed. Lines (P068) – (P070) require the body of the output message must be encrypted.

An example of an input message that conforms to the above stated policy is as follows.

```
(M001)   <?xml version="1.0" encoding="utf-8" ?>
(M002)   <soap:Envelope xmlns:soap="..." xmlns:xenc="..." xmlns:ds="...">
(M003)     <soap:Header>
(M004)       <wsse:Security soap:mustUnderstand="1" xmlns:wsse="..."
xmlns:wsu="...">
(M005)         <xenc:EncryptedKey wsu:Id="EK">
(M006)           <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
(M007)            <ds:KeyInfo>
(M008)           <wsse:SecurityTokenReference>
(M009)             <wsse:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/oasis-wss-soap-message-security-1.1#ThumbPrintSHA1">
(M010)               LKiQ/CmFrJDJqCLFcjlhIsmZ/+0=
(M011)             </wsse:KeyIdentifier>
(M012)           </wsse:SecurityTokenReference>
(M013)         </ds:KeyInfo>
(M014)         </xenc:EncryptedKey>
```

```
(M015)          <xenc:ReferenceList>
(M016)            <xenc:DataReference URI=”#encUT”/>
(M017)            <xenc:DataReference URI=”#encBody”/>
(M018)          </xenc:ReferenceList>
(M019)          <wsu:Timestamp wsu:Id="T0">
(M020)            <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
(M021)          </wsu:Timestamp>
(M022)          <xenc:EncryptedData wsu:Id="encUT">
(M023)            <xenc:EncryptionMethod
Algorithm=”http://www.w3.org/2001/04/xmlenc#aes256-cbc”/>
(M024)            <ds:KeyInfo>
(M025)              <wsse:SecurityTokenReference>
(M026)                <wsse:Reference URI=”#EK”/>
(M027)              </wsse:SecurityTokenReference>
(M028)            </ds:KeyInfo>
(M029)            <xenc:CipherData>
(M030)              <xenc:CipherValue>...</xenc:CipherValue>
(M031)            </xenc:CipherData>
(M032)          </xenc:EncryptedData>
(M033)          <ds:Signature>
(M034)            <ds:SignedInfo>…
(M035)              <ds:Reference URI="#T0">…</ds:Reference>
(M036)              <ds:Reference URI="#usernameToken">…</ds:Reference>
(M037)              <ds:Reference URI="#body">…</ds:Reference>
(M038)            </ds:SignedInfo>
(M039)            <ds:SignatureValue>HFLP…</ds:SignatureValue>
(M040)            <ds:KeyInfo>
(M041)              <wsse:SecurityTokenReference>
(M042)                <wsse:Reference URI=”#EK”/>
(M043)              </wsse:SecurityTokenReference>
(M044)            </ds:KeyInfo>
(M045)          </ds:Signature>
(M046)        </wsse:Security>
(M047)      </soap:Header>
(M048)      <soap:Body wsu:Id="body">
(M049)        <xenc:EncryptedData wsu:Id="encBody">
(M050)          <xenc:EncryptionMethod
Algorithm=”http://www.w3.org/2001/04/xmlenc#aes256-cbc”/>
(M051)          <ds:KeyInfo>
(M052)            <wsse:SecurityTokenReference>
(M053)              <wsse:Reference URI=”#EK”/>
(M054)            </wsse:SecurityTokenReference>
(M055)          </ds:KeyInfo>
(M056)          <xenc:CipherData>
(M057)            <xenc:CipherValue>...</xenc:CipherValue>
(M058)          </xenc:CipherData>
(M059)        </xenc:EncryptedData>
(M060)      </soap:Body>
(M061)    </soap:Envelope>
```

Lines (M005) – (M014) contain the encrypted symmetric key as required by the use of the SymmetricBinding assertion staring on line (P004) with an X509Token ProtectionToken assertion. Line (M006) references the KwRsaOaep Asymmetric Key Wrap algorithm dictated by the Basic 256 Algorithm Suite assertion on line (P018). Lines (M009) – (M011) hold a KeyIdentifier of the protection token used to encrypt the symmetric key as required by the SymmetricBinding assertion. Because the ProtectionToken assertion disallowed the token from being inserted into the message and instead required a thumbprint reference, a thumbprint reference is included to identify the token.

Line (M016) is an encryption data reference that references the encrypted supporting UsernameToken on lines (M022) – (M032). The encryption was required by the SignedEncryptedSupportingTokens assertion on line (P038). Line (M023) references the Aes256 Encryption algorithm dictated by the Basic 256 Algorithm Suite assertion on line (P018). The encrypted symmetric key is used to encrypt the UsernameToken as referenced on line (M026).

Line (M017) is an encryption data reference that references the encrypted body of the message on lines (M049) – (M059). The encryption was required by the EncryptedParts assertion of the input message policy. The encrypted symmetric key is used to encrypt the UsernameToken as referenced on line (M053).

Lines (M019) – (M021) contain a timestamp for the message as required by the IncludeTimestamp assertion.

Lines (M033) – (M045) contain the message signature.

Line (M035) indicates the message timestamp is included in the signature as required by the IncludeTimestamp assertion definition.

Line (M036) indicates the supporting UsernameToken is included in the signature as required by the SignedSupportingTokens assertion. Because the token was encrypted its content prior to encryption is included below to better illustrate the reference.

```
(M062)    <wsse:UsernameToken wsu:Id="usernameToken">
(M063)      <wsse:Username>Chris</wsse:Username>
(M064)      <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-1.0#PasswordText ">sirhC</wsse:Password>
(M065)    </wsse:UsernameToken>
```

Line (M037) indicates the message body is included in the signature as required by the SignedParts assertion of the input message policy.

Line (M042) references the encrypted symmetric key for signing as dictated by the SymmetricBinding assertion.

2.2 X.509 Token Authentication Scenario Assertions

2.2.1 (WSS1.0) X.509 Certificates, Sign, Encrypt

This use-case corresponds to the situation where both parties have X.509v3 certificates (and public-private key pairs). The requestor identifies itself to the service. The message exchange is integrity protected and encrypted.

This modeled by use of an asymmetric security binding assertion.

The message level policies in this and subsequent sections cover a different scope of the web service definition than the security binding level policy and so appear as separate policies and are attached at WSDL Message Policy Subject. These are shown below as input and output policies. Thus, we need a set of coordinated policies one with endpoint subject and two with message subjects to achieve this use case.

The policies are as follows:

```
(P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >
(P002)   <wsp:ExactlyOne>
(P003)     <wsp:All>
(P004)       <sp:AsymmetricBinding>
(P005)         <wsp:Policy>
(P006)           <sp:InitiatorToken>
(P007)             <wsp:Policy>
(P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P009)                 <wsp:Policy>
(P010)                   <sp:WssX509V3Token10/>
(P011)                 </wsp:Policy>
(P012)               </sp:X509Token>
(P013)             </wsp:Policy>
(P014)           </sp:InitiatorToken>
(P015)           <sp:RecipientToken>
(P016)             <wsp:Policy>
(P017)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
sx/ws-securitypolicy/200702/IncludeToken/Never">
```

```
(P018)                 <wsp:Policy>
(P019)                   <sp:WssX509V3Token10/>
(P020)                 </wsp:Policy>
(P021)               </sp:X509Token>
(P022)             </wsp:Policy>
(P023)           </sp:RecipientToken>
(P024)           <sp:AlgorithmSuite>
(P025)             <wsp:Policy>
(P026)               <sp:Basic256/>
(P027)             </wsp:Policy>
(P028)           </sp:AlgorithmSuite>
(P029)           <sp:Layout>
(P030)             <wsp:Policy>
(P031)               <sp:Strict/>
(P032)             </wsp:Policy>
(P033)           </sp:Layout>
(P034)           <sp:IncludeTimestamp/>
(P035)           <sp:OnlySignEntireHeadersAndBody/>
(P036)         </wsp:Policy>
(P037)       </sp:AsymmetricBinding>
(P038)       <sp:Wss10>
(P039)         <wsp:Policy>
(P040)           <sp:MustSupportRefKeyIdentifier/>
(P041)         </wsp:Policy>
(P042)       </sp:Wss10>
(P043)     </wsp:All>
(P044)   </wsp:ExactlyOne>
(P045) </wsp:Policy>

(P046) <wsp:Policy wsu:Id="WSS10Anonymous_with_Certificates_input_policy">
(P047)   <wsp:ExactlyOne>
(P048)     <wsp:All>
(P049)       <sp:SignedParts>
(P050)         <sp:Body/>
(P051)       </sp:SignedParts>
(P052)       <sp:EncryptedParts>
(P053)         <sp:Body/>
(P054)       </sp:EncryptedParts>
(P055)     </wsp:All>
(P056)   </wsp:ExactlyOne>
(P057) </wsp:Policy>

(P058) <wsp:Policy wsu:Id="WSS10anonymous_with_certs_output_policy">
(P059)   <wsp:ExactlyOne>
(P060)     <wsp:All>
(P061)       <sp:SignedParts>
(P062)         <sp:Body/>
(P063)       </sp:SignedParts>
(P064)       <sp:EncryptedParts>
(P065)         <sp:Body/>
(P066)       </sp:EncryptedParts>
(P067)     </wsp:All>
(P068)   </wsp:ExactlyOne>
(P069) </wsp:Policy>
```

Lines (P004) – (P037) contain the AsymmetricBinding assertion which indicates that the initiator's token must be used for the message signature and the recipient's token must be used for message encryption.

Lines (P006) – (P014) contain the InitiatorToken assertion. Within that assertion lines (P008) – (P012) indicate that the initiator token must be an X.509 token that must be included with all messages sent to the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as described in the WS-Security 1.0 X.509 Token Profile.

Lines (P015) – (P023) contain the RecipientToken assertion. Within that assertion lines (P017) – (P021) dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509 Token

Profile, however as stated on line (P017) it must not be included in any message. Instead, according to the MustSupportKeyRefIdentitier assertion on line (P040) a KeyIdentifier must be used to identify the token in any messages where the token is used.

Line (P034) requires the inclusion of a timestamp.

Lines (P046) – (P057) contain a policy that is attached to the input message. Lines (P049) – (P051) require that the body of the input message must be signed. Lines (P052) – (P054) require the body of the input message must be encrypted.

Lines (P058) – (P069) contain a policy that is attached to the output message. Lines (P061) – (P063) require that the body of the output message must be signed. Lines (P064) – (P066) require the body of the output message must be encrypted.

An example of an input message that conforms to the above stated policy is as follows.

```
(M001) <?xml version="1.0" encoding="utf-8" ?>
(M002) <soap:Envelope xmlns:soap="..." xmlns:xenc="..." xmlns:ds="...">
(M003)   <soap:Header>
(M004)     <wsse:Security soap:mustUnderstand="1" xmlns:wsse="..." xmlns:wsu="...">
(M005)       <xenc:EncryptedKey >
(M006)         <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
(M007)           <ds:DigestMethod Algorithm="...#sha1"/>
(M008)         </xenc:EncryptionMethod>
(M009)         <ds:KeyInfo>
(M010)           <wsse:SecurityTokenReference >
(M011)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
ValueType="...#X509SubjectKeyIdentifier">
(M012)             MIGfMa0GCSq…
(M013)             </wsse:KeyIdentifier>
(M014)           </ds:KeyInfo>
(M015)         <xenc:CipherData>
(M016)           <xenc:CipherValue>Hyx...=</xenc:CipherValue>
(M017)         </xenc:CipherData>
(M018)         <xenc:ReferenceList>
(M019)           <xenc:DataReference URI="#encBody"/>
(M020)         </xenc:ReferenceList>
(M021)       </xenc:EncryptedKey>
(M022)       <wsu:Timestamp wsu:Id="T0">
(M023)         <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
(M024)       </wsu:Timestamp>
(M025)       <wsse:BinarySecurityToken wsu:Id="binaryToken" ValueType="…#X509v3"
EncodingType="…#Base64Binary">
(M026)         MIIEZzCCA9CgAwIBAgIQEmtJZc0…
(M027)       </wsse:BinarySecurityToken>
(M028)       <ds:Signature>
(M029)         <ds:SignedInfo>…
(M030)           <ds:Reference URI="#T0">…</ds:Reference>
(M031)           <ds:Reference URI="#body">…</ds:Reference>
(M032)         </ds:SignedInfo>
(M033)         <ds:SignatureValue>HFLP…</ds:SignatureValue>
(M034)         <ds:KeyInfo>
(M035)           <wsse:SecurityTokenReference>
(M036)             <wsse:Reference URI="#binaryToken"/>
(M037)           </wsse:SecurityTokenReference>
(M038)         </ds:KeyInfo>
(M039)       </ds:Signature>
(M040)     </wsse:Security>
(M041)   </soap:Header>
(M042)   <soap:Body wsu:Id="body">
(M043)     <xenc:EncryptedData wsu:Id="encBody">
(M044)       <xenc:CipherData>
(M045)         <xenc:CipherValue>...</xenc:CipherValue>
(M046)       </xenc:CipherData>
(M047)     </xenc:EncryptedData>
(M048)   </soap:Body>
```

```
(M049) </soap:Envelope>
```

Line (M019) is an encryption data reference that references the encrypted body of the message on lines (M043) – (M047). The encryption was required by the EncryptedParts assertion of the input message policy. Lines (M011) – (M013) hold a KeyIdentifier of the recipient’s token used to encrypt the body as required by the AsymmetricBinding assertion. Because the RecipientToken assertion disallowed the token from being inserted into the message, a KeyIdentifier is used instead of a reference to an included token.

Lines (M022) – (M024) contain a timestamp for the message as required by the IncludeTimestamp assertion.

Lines (M025) – (M027) contain the BinarySecurityToken holding the X.509v3 certificate of the initiator as required by the InitiatorToken assertion.

Lines (M028) – (M039) contain the message signature.

Line (M030) indicates the message timestamp is included in the signature as required by the IncludeTimestamp assertion definition.

Line (M031) indicates the message body is included in the signature as required by the SignedParts assertion of the input message policy.

Note that the initiator’s BinarySecurityToken is not included in the message signature as it was not required by policy.

Lines (M035) – (M037) references the initiator’s BinarySecurityToken included in the message for identifying the key used for signing as dictated by the AsymmetricBinding assertion.

2.2.2 (WSS1.0) Mutual Authentication with X.509 Certificates, Sign, Encrypt

This scenario is based on WSS Interop, Scenario 3, Web Services Security: Interop 1, Draft 06, Editor, Hal Lockhart, BEA Systems

This use case corresponds to the situation where both parties have X.509v3 certificates (and public-private key pairs). The requestor wishes to identify itself to the service using its X.509 credential (strong authentication). The message exchange needs to be integrity protected and encrypted as well. The difference from previous use case is that the X509 token inserted by the client is included in the message signature (see <ProtectTokens />).

The policy is as follows:

```
(P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >
(P002)   <wsp:ExactlyOne>
(P003)     <wsp:All>
(P004)       <sp:AsymmetricBinding>
(P005)         <wsp:Policy>
(P006)           <sp:InitiatorToken>
(P007)             <wsp:Policy>
(P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P009)                 <wsp:Policy>
(P010)                   <sp:WssX509V3Token10/>
(P011)                 </wsp:Policy>
(P012)               </sp:X509Token>
(P013)             </wsp:Policy>
(P014)           </sp:InitiatorToken>
(P015)           <sp:RecipientToken>
(P016)             <wsp:Policy>
(P017)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
sx/ws-securitypolicy/200702/IncludeToken/Never">
(P018)                 <wsp:Policy>
(P019)                   <sp:WssX509V3Token10/>
(P020)                 </wsp:Policy>
(P021)               </sp:X509Token>
(P022)             </wsp:Policy>
(P023)           </sp:RecipientToken>
```

```
(P024)           <sp:AlgorithmSuite>
(P025)             <wsp:Policy>
(P026)               <sp:Basic256/>
(P027)             </wsp:Policy>
(P028)           </sp:AlgorithmSuite>
(P029)           <sp:Layout>
(P030)             <wsp:Policy>
(P031)               <sp:Strict/>
(P032)             </wsp:Policy>
(P033)           </sp:Layout>
(P034)           <sp:IncludeTimestamp/>
(P035)           <sp:ProtectTokens />
(P036)           <sp:OnlySignEntireHeadersAndBody/>
(P037)         </wsp:Policy>
(P038)       </sp:AsymmetricBinding>
(P039)       <sp:Wss10>
(P040)         <wsp:Policy>
(P041)           <sp:MustSupportRefKeyIdentifier/>
(P042)         </wsp:Policy>
(P043)       </sp:Wss10>
(P044)     </wsp:All>
(P045)   </wsp:ExactlyOne>
(P046) </wsp:Policy>

(P047) <wsp:Policy wsu:Id="WSS10Anonymous with Certificates_input_policy">
(P048)   <wsp:ExactlyOne>
(P049)     <wsp:All>
(P050)       <sp:SignedParts>
(P051)         <sp:Body/>
(P052)       </sp:SignedParts>
(P053)       <sp:EncryptedParts>
(P054)         <sp:Body/>
(P055)       </sp:EncryptedParts>
(P056)     </wsp:All>
(P057)   </wsp:ExactlyOne>
(P058) </wsp:Policy>

(P059) <wsp:Policy wsu:Id="WSS10anonymous with certs_output_policy">
(P060)   <wsp:ExactlyOne>
(P061)     <wsp:All>
(P062)       <sp:SignedParts>
(P063)         <sp:Body/>
(P064)       </sp:SignedParts>
(P065)       <sp:EncryptedParts>
(P066)         <sp:Body/>
(P067)       </sp:EncryptedParts>
(P068)     </wsp:All>
(P069)   </wsp:ExactlyOne>
(P070) </wsp:Policy>
```

Lines (P004) – (P038) contain the AsymmetricBinding assertion which indicates that the initiator's token must be used for the message signature and the recipient's token must be used for message encryption.

Lines (P006) – (P014) contain the InitiatorToken assertion. Within that assertion lines (P008) – (P012) indicate that the initiator token must be an X.509 token that must be included with all messages sent to the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as described in the WS-Security 1.0 X.509 Token Profile.

Lines (P015) – (P023) contain the RecipientToken assertion. Within that assertion lines (P017) – (P021) dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509 Token Profile, however as stated on line (P017) it must not be included in any message. Instead, according to the MustSupportKeyRefIdentitier assertion on line (P040) a KeyIdentifier must be used to identify the token in any messages where the token is used.

Line (P034) requires the inclusion of a timestamp.

Line (P035) requires token protection (ProtectTokens) which dictates that the signature must cover the token used to generate that signature.

Lines (P047) – (P058) contain a policy that is attached to the input message. Lines (P050) – (P052) require that the body of the input message must be signed. Lines (P053) – (P055) require the body of the input message must be encrypted.

Lines (P059) – (P070) contain a policy that is attached to the output message. Lines (P062) – (P064) require that the body of the output message must be signed. Lines (P064) – (P066) require the body of the output message must be encrypted.

An example of an input message that conforms to the above stated policy is as follows.

```
(M001) <?xml version="1.0" encoding="utf-8" ?>
(M002) <soapenv:Envelope  xmlns:soapenv="..."
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsu="..." xmlns:xenc="..." ...>
(M003)   <soapenv:Header>
(M004)     <wsse:Security xmlns:wsse="..." xmlns:ds="..."
soapenv:mustUnderstand="1">
(M005)       <xenc:EncryptedKey >
(M006)         <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
(M007)           <ds:DigestMethod Algorithm="...#sha1"/>
(M008)         </xenc:EncryptionMethod>
(M009)         <ds:KeyInfo>
(M010)           <wsse:SecurityTokenReference >
(M011)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
ValueType="...#X509SubjectKeyIdentifier">CuJd...=</wsse:KeyIdentifier>
(M012)           </wsse:SecurityTokenReference>
(M013)         </ds:KeyInfo>
(M014)         <xenc:CipherData>
(M015)           <xenc:CipherValue>Hyx...=</xenc:CipherValue>
(M016)         </xenc:CipherData>
(M017)         <xenc:ReferenceList>
(M018)           <xenc:DataReference URI="#encBody"/>
(M019)         </xenc:ReferenceList>
(M020)       </xenc:EncryptedKey>
(M021)       <wsu:Timestamp wsu:Id="Timestamp" >
(M022)         <wsu:Created>2007-03-26T16:53:39Z</wsu:Created>
(M023)       </wsu:Timestamp>
(M024)       <wsse:BinarySecurityToken wsu:Id="bst" ValueType="...#X509v3"
EncodingType="...#Base64Binary">MIID...=</wsse:BinarySecurityToken>
(M025)       <ds:Signature>
(M026)         <ds:SignedInfo>
(M027)           <ds:CanonicalizationMethod Algorithm=".../xml-exc-c14n#"/>
(M028)           <ds:SignatureMethod Algorithm="...#rsa-sha1"/>
(M029)           <ds:Reference URI="#Timestamp">
(M030)             <ds:Transforms>... </ds:Transforms>
(M031)             <ds:DigestMethod Algorithm="...#sha1"/>
(M032)             <ds:DigestValue>+g0I...=</ds:DigestValue>
(M033)           </ds:Reference>
(M034)           <ds:Reference URI="#Body">...</ds:Reference>
(M035)           <ds:Reference URI="#bst">..</ds:Reference>
(M036)         </ds:SignedInfo>
(M037)         <ds:SignatureValue>RRT...=</ds:SignatureValue>
(M038)         <ds:KeyInfo>
(M039)           <wsse:SecurityTokenReference >
(M040)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
ValueType="...#X509SubjectKeyIdentifier">Xeg5...=</wsse:KeyIdentifier>
(M041)           </wsse:SecurityTokenReference>
(M042)         </ds:KeyInfo>
(M043)       </ds:Signature>
(M044)     </wsse:Security>
(M045)   </soapenv:Header>
(M046)   <soapenv:Body wsu:Id="Body" >
(M047)     <xenc:EncryptedData Id="encBody" Type="...#Content"
(M048)        MimeType="text/xml" Encoding="UTF-8" >
```

```
(M049)        <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
(M050)        <xenc:CipherData>
(M051)          <xenc:CipherValue>W84fn...1</xenc:CipherValue>
(M052)        </xenc:CipherData>
(M053)      </xenc:EncryptedData>
(M054)    </soapenv:Body>
(M055) </soapenv:Envelope>
```

Line (M018) is an encryption data reference that references the encrypted body of the message on lines (M047) – (M048). The encryption was required by the EncryptedParts assertion of the input message policy. Lines (M009) – (M013) hold a KeyIdentifier of the recipient's token used to encrypt the body as required by the AsymmetricBinding assertion. Because the RecipientToken assertion disallowed the token from being inserted into the message, a KeyIdentifier is used instead of a reference to an included token.

Lines (M021) – (M023) contain a timestamp for the message as required by the IncludeTimestamp assertion.

Line (M024) contains the BinarySecurityToken holding the X.509v3 certificate of the initiator as required by the InitiatorToken assertion.

Lines (M025) – (M043) contain the message signature.

Line (M029) indicates the message timestamp is included in the signature as required by the IncludeTimestamp assertion definition.

Line (M034) indicates the message body is included in the signature as required by the SignedParts assertion of the input message policy.

Line (M035) indicates the BinarySecurityToken on Line (M024) is included in the signature as required by the ProtectTokens assertion of the AsymmetricBinding assertion policy.

Note that the recipient's token is not explicitly included in the security header, as it is required not to be by policy (P017). Instead a KeyIdentifier, line (M011) is used to identify the recipient's that should be used to decrypt the EncryptedData.

Lines (M039) – (M041) reference the initiator's BinarySecurityToken on line (M034), which is included in the message to contain the key used for verifying as dictated by the AsymmetricBinding assertion.

2.2.2.1 (WSS1.0) Mutual Authentication, X.509 Certificates, Symmetric Encryption

This scenario is based on WSS Interop, Scenario 4, Web Services Security: Interop 2.

A common variation on the previous example is where X.509 Certificates are still used for authentication, but a mutually agreed upon symmetric key is used for encryption.

In this use case the policy is the same except that the InitiatorToken and RecipientToken are now each just SignatureTokens.

A second variation in this use case is that the mutually agreed upon symmetric encryption key is characterized as an "IssuedToken" with a mutually agreed upon URI used to identify the out of band (not included in the message) token, which is included in a SupportingTokens element.

The policy is as follows:

```
(P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >
(P002)   <wsp:ExactlyOne>
(P003)     <wsp:All>
(P004)       <sp:AsymmetricBinding>
(P005)         <wsp:Policy>
(P006)           <sp:InitiatorSignatureToken>
(P007)             <wsp:Policy>
(P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P009)                 <wsp:Policy>
(P010)                   <sp:WssX509V3Token10/>
(P011)                 </wsp:Policy>
(P012)               </sp:X509Token>
(P013)             </wsp:Policy>
```

```
(P014)          </sp:InitiatorSignatureToken>
(P015)          <sp:RecipientSignatureToken>
(P016)            <wsp:Policy>
(P017)              <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
sx/ws-securitypolicy/200702/IncludeToken/Never">
(P018)                <wsp:Policy>
(P019)                  <sp:WssX509V3Token10/>
(P020)                </wsp:Policy>
(P021)              </sp:X509Token>
(P022)            </wsp:Policy>
(P023)          </sp:RecipientSignatureToken>
(P024)          <sp:AlgorithmSuite>
(P025)            <wsp:Policy>
(P026)              <sp:Basic256/>
(P027)            </wsp:Policy>
(P028)          </sp:AlgorithmSuite>
(P029)          <sp:Layout>
(P030)            <wsp:Policy>
(P031)              <sp:Strict/>
(P032)            </wsp:Policy>
(P033)          </sp:Layout>
(P034)          <sp:IncludeTimestamp/>
(P035)          <sp:ProtectTokens />
(P036)          <sp:OnlySignEntireHeadersAndBody/>
(P037)        </wsp:Policy>
(P038)      </sp:AsymmetricBinding>
(P039)      <sp:SupportingTokens>
(P040)        <wsp:Policy>
(P041)          <sp:IssuedToken>
(P042)            <sp:Issuer>SomeMutuallyAgreedURI</sp:Issuer>
(P043)            <sp:RequireExternalReference/>
(P044)          </sp:IssuedToken>
(P045)          <sp:EncryptedParts>
(P046)             <sp:Body/>
(P047)          </sp:EncryptedParts>
(P048)        </wsp:Policy>
(P049)      </sp:SupportingTokens>
(P050)      <sp:Wss10>
(P051)        <wsp:Policy>
(P052)          <sp:MustSupportRefKeyIdentifier/>
(P053)        </wsp:Policy>
(P054)      </sp:Wss10>
(P055)    </wsp:All>
(P056)  </wsp:ExactlyOne>
(P057) </wsp:Policy>

(P058) <wsp:Policy wsu:Id="WSS10Anonymous with Certificates_input_policy">
(P059)   <wsp:ExactlyOne>
(P060)     <wsp:All>
(P061)       <sp:SignedParts>
(P062)         <sp:Body/>
(P063)       </sp:SignedParts>
(P064)       <sp:EncryptedParts>
(P065)         <sp:Body/>
(P066)       </sp:EncryptedParts>
(P067)     </wsp:All>
(P068)   </wsp:ExactlyOne>
(P069) </wsp:Policy>

(P070) <wsp:Policy wsu:Id="WSS10anonymous with certs_output_policy">
(P071)   <wsp:ExactlyOne>
(P072)     <wsp:All>
(P073)       <sp:SignedParts>
(P074)         <sp:Body/>
```

```
(P075)       </sp:SignedParts>
(P076)       <sp:EncryptedParts>
(P077)         <sp:Body/>
(P078)       </sp:EncryptedParts>
(P079)     </wsp:All>
(P080)   </wsp:ExactlyOne>
(P081) </wsp:Policy>
```

Lines (P004) – (P038) contain the AsymmetricBindingAssertion which indicates that the initiator's token must be used for the message signature and the recipient's token must be used for message encryption.

Lines (P006) – (P014) contain the InitiatorSignatureToken assertion. Within that assertion lines (P008) – (P012) indicate that the initiator token must be an X.509 token that must be included with all messages sent to the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as described in the WS-Security 1.0 X.509 Token Profile. By specifying that this is an InitiatorSignatureToken, it will only be used to sign the message and not used for encryption.

Lines (P015) – (P023) contain the RecipientSignatureToken assertion. Within that assertion lines (P017) – (P021) dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509 Token Profile, however as stated on line (P017) it must not be included in any message. Instead, according to the MustSupportKeyRefIdentitier assertion on line (P040) a KeyIdentifier must be used to identify the token in any messages where the token is used. By specifying that this is an RecipientSignatureToken, it will only be used to sign the response and not used for encryption.

Line (P034) requires the inclusion of a timestamp.

Line (P035) requires token protection (ProtectTokens) which dictates that the signature must cover the token or token reference associated with the generation of that signature.

Lines (P039) – (P049) contain a SupportingTokens assertion that contains an IssuedToken (P041) – (P044), which contains an Issuer element (P042) that identifies an explicit URI ("SomeMutuallyAgreedURI") that must be used to indicate what token will be used. (Since the content of the IssuedToken element is specific to the Issuer, any mechanism can be used to identify the key, and in this case the simplest method of identifying the issuer with the explicit key has been chosen only to illustrate one possible method, but not to recommend as opposed to any other method.) Line (P043), RequireExternalReference indicates that the IssuedToken requires a token that is referenced external to the message. Lines (P045) – (P047) indicate that the token is to be used for encrypting parts of the message, explicitly, line (P046), the Body of the message.

Lines (P058) – (P069) contain a policy that is attached to the input message. Lines (P061) – (P063) require that the body of the input message must be signed. Lines (P064) – (P066) require the body of the input message must be encrypted.

Lines (P070) – (P081) contain a policy that is attached to the output message. Lines (P073) – (P075) require that the body of the output message must be signed. Lines (P076) – (P078) require the body of the output message must be encrypted.

The following example message is derived from the WSS Interop2 document.

```
(M001)    <?xml version="1.0" encoding="utf-8" ?>
(M002)    <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
(M003)      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
(M004)      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
(M005)     <soap:Header>
(M006)      <wsse:Security soap:mustUnderstand="1"
(M007)        xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
(M008)       <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
(M009)         <xenc:DataReference URI="#enc" />
(M010)       </xenc:ReferenceList>
(M011)       <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
(M012)          wsu:Id="timestamp">
(M013)        <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
(M014)       </wsu:Timestamp>
(M015)       <wsse:BinarySecurityToken ValueType="wsse:X509v3"
(M016)         EncodingType="wsse:Base64Binary"
```

```
(M017)         xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
(M018)         wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
(M019)       <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
(M020)        <SignedInfo>
(M021)         <CanonicalizationMethod
(M022)           Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
(M023)         <SignatureMethod
(M024)           Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
(M025)         <Reference URI="#body">
(M026)          <Transforms>
(M027)           <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M028)          </Transforms>
(M029)          <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(M030)          <DigestValue>QTV...dw=</DigestValue>
(M031)         </Reference>
(M032)         <Reference URI="#myCert">
(M033)          <Transforms>
(M034)           <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M035)          </Transforms>
(M036)          <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(M037)          <DigestValue>XYZ...ab=</DigestValue>
(M038)         </Reference>
(M039)         <Reference URI="#timestamp">
(M040)          <Transforms>
(M041)           <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M042)          </Transforms>
(M043)          <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(M044)          <DigestValue>XYZ...ab=</DigestValue>
(M045)         </Reference>
(M046)        </SignedInfo>
(M047)        <SignatureValue>H+x0...gUw=</SignatureValue>
(M048)        <KeyInfo>
(M049)         <wsse:SecurityTokenReference>
(M050)          <wsse:Reference URI="#myCert" />
(M051)         </wsse:SecurityTokenReference>
(M052)        </KeyInfo>
(M053)       </Signature>
(M054)      </wsse:Security>
(M055)     </soap:Header>
(M056)     <soap:Body wsu:Id="body"
(M057)       xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
(M058)      <xenc:EncryptedData Id="enc"
(M059)        Type="http://www.w3.org/2001/04/xmlenc#Content"
(M060)        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
(M061)       <xenc:EncryptionMethod
(M062)         Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
(M063)       <xenc:KeyInfo>
(M064)        <xenc:KeyName>SomeMutuallyAgreedURI</xenc:Keyname>
(M065)       </xenc:KeyInfo>
(M066)       <xenc:CipherData>
(M067)        <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
(M068)       </xenc:CipherData>
(M069)      </xenc:EncryptedData>
(M070)     </soap:Body>
(M071)    </soap:Envelope>
```

In the above example, the main point of attention is line (M064), where the KeyName of the EncryptedData element is “SomeMutuallyAgreedURI”. As indicated above the specific techniques used to identify and apply this token are totally within agreed upon methods define by the mutual parties.

Lines (M005) – (M055) contain the SOAP Header element, which contains the WS-Security header (M006) – (M054).

The ReferenceList (M008) – (M010) contains an internal reference (M009), “enc”, identifying the EncryptedData element Id, line (M058). This EncryptedData (M058) – (M069) inside the SOAP Body, (M056) – (M070), was required to be encrypted by the policy (P058) – (P069) as described above.

The Timestamp (M011) – (M014) is required by the policy line (P034).

The BinarySecurityToken (M015) – (M018) contains the actual certificate used to sign the request as required by the policy, (P008), IncludeToken/AlwaysToRecipient.

The Signature (M019) – (M053) contains a SignedInfo (M020) – (M046) that identifies the “#body” (M025) the “#myCert” (M032), and the “#timestamp” (M039) as the elements covered by the signature. The Reference (M032) – (M038) with URI=”#myCert” that covers the BinarySecurityToken (M015) – (M018) was required by ProtectTokens in the policy (P035).

The KeyInfo (M048) – (M052) uses a SecurityTokenReference to point to the “myCert” BinarySecurityToken.

The EncryptedData (M058) – (M069) was described above, where it was pointed out that line (M064) contains the external reference (“SomeMutuallyAgreedURI”) to the mutually shared symmetric key used for encryption.

2.2.3 (WSS1.1) Anonymous with X.509 Certificate, Sign, Encrypt

This scenario is based on the the “Examples of Secure Web Service Message Exchange Document” [WS-SECURE-INTEROP] (see also sec 2.1.4)

In this use case the Request is signed using DerivedKeyToken1(K), then encrypted using a DerivedKeyToken2(K) where K is ephemeral key protected for the server's certificate. Response is signed using DKT3(K), (if needed) encrypted using DKT4(K). The requestor does not wish to identify himself; the message exchange is protected using derived symmetric keys. As a simpler, but less secure, alternative, ephemeral key K (instead of derived keys) could be used for message protection by simply omitting the sp:RequireDerivedKeys assertion.

The policy is as follows:

```
(P001) <wsp:Policy wsu:Id="WSS11_AnonymousForX509SignEncrypt_Policy">
(P002)   <wsp:ExactlyOne>
(P003)     <wsp:All>
(P004)       <sp:SymmetricBinding>
(P005)         <wsp:Policy>
(P006)           <sp:ProtectionToken>
(P007)             <wsp:Policy>
(P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
sx/ws-securitypolicy/200702/IncludeToken/Never">
(P009)                 <wsp:Policy>
(P010)                   <sp:RequireDerivedKeys/>
(P011)                   <sp:RequireThumbprintReference/>
(P012)                   <sp:WssX509V3Token11/>
(P013)                 </wsp:Policy>
(P014)               </sp:X509Token>
(P015)             </wsp:Policy>
(P016)           </sp:ProtectionToken>
(P017)           <sp:AlgorithmSuite>
(P018)             <wsp:Policy>
(P019)               <sp:Basic256/>
(P020)             </wsp:Policy>
(P021)           </sp:AlgorithmSuite>
(P022)           <sp:Layout>
(P023)             <wsp:Policy>
(P024)               <sp:Strict/>
(P025)             </wsp:Policy>
(P026)           </sp:Layout>
(P027)           <sp:IncludeTimestamp/>
(P028)           <sp:OnlySignEntireHeadersAndBody/>
(P029)         </wsp:Policy>
(P030)       </sp:SymmetricBinding>
(P031)       <sp:Wss11>
(P032)         <wsp:Policy>
(P033)           <sp:MustSupportRefKeyIdentifier/>
(P034)           <sp:MustSupportRefIssuerSerial/>
(P035)           <sp:MustSupportRefThumbprint/>
(P036)           <sp:MustSupportRefEncryptedKey/>
(P037)           <sp:RequireSignatureConfirmation/>
(P038)         </wsp:Policy>
```

```
(P039)       </sp:Wss11>
(P040)     </wsp:All>
(P041)   </wsp:ExactlyOne>
(P042) </wsp:Policy>

(P043) <wsp:Policy wsu:Id=" WSS11_AnonymousForX509SignEncrypt_input_policy">
(P044)   <wsp:ExactlyOne>
(P045)     <wsp:All>
(P046)       <sp:SignedParts>
(P047)         <sp:Body/>
(P048)       </sp:SignedParts>
(P049)       <sp:EncryptedParts>
(P050)         <sp:Body/>
(P051)       </sp:EncryptedParts>
(P052)     </wsp:All>
(P053)   </wsp:ExactlyOne>
(P054) </wsp:Policy>

(P055) <wsp:Policy wsu:Id=" WSS11_AnonymousForX509SignEncrypt_output_policy">
(P056)   <wsp:ExactlyOne>
(P057)     <wsp:All>
(P058)       <sp:SignedParts>
(P059)         <sp:Body/>
(P060)       </sp:SignedParts>
(P061)       <sp:EncryptedParts>
(P062)         <sp:Body/>
(P063)       </sp:EncryptedParts>
(P064)     </wsp:All>
(P065)   </wsp:ExactlyOne>
(P066) </wsp:Policy>
```

Lines (P004) – (P030) contain the SymmetricBinding assertion which indicates that the derived key token must be used for both message signature and message encryption.

Lines (P007) – (P016) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P014) indicate that the ProtectionToken must be an X.509 token that MUST NOT be included with any message sent between the Initiator and Recipient.

Line (P010) dictates the derived key is required. Line (P012) dictates the X.509 token must be an X.509v3 security token as described in the WS-Security 1.1 X.509 Token Profile. According to the MustSupportRefThumbprint assertion on line (P035) and RequireThumbprintReference on line (P011), a Thumbprint Reference of KeyIdentifier must be used to identify the token in any messages where the token is used.

Line (P027) requires the inclusion of a timestamp.

Lines (P031) – (P039) contain some WS-Security 1.1 related interoperability requirements, specifically support for key identifier, issuer serial number, thumbprint, encrypted key references, and requires signature confirmation on the response.

Lines (P043) – (P054) contain a policy that is attached to the input message. Lines (P045) – (P048) require that the body of the input message must be signed. Lines (P049) – (P051) require the body of the input message must be encrypted.

Lines (P055) – (P066) contain a policy that is attached to the output message. Lines (P057) – (P060) require that the body of the output message must be signed. Lines (P061) – (P063) require the body of the output message must be encrypted.

An example of an input message that conforms to the above stated policy is as follows.

Note: this message uses WS-SecureConversation as a means to meet the requirements of the policy, however, aside from using the wsc:DerivedKeyToken elements to meet the policy requirements for the RequireDerivedKeys assertion (P010) the general protocol mechanisms described in WS-SecureConversation for SecurityContextTokens are not explicitly demonstrated.

```
(M001)  <?xml version="1.0" encoding="utf-8" ?>
(M002)  <env:Envelope xmlns:env="..." xmlns:xenc="http..." xmlns:ds="..." xmlns:wsu="...">
```

```
(M003)  <env:Header>
(M004)   <wsse:Security xmlns:wsse="..." xmlns:wsse11="..." xmlns:wsc="..."
env:mustUnderstand="1">
(M005)    <xenc:EncryptedKey Id="encKey" >
(M006)      <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
(M007)        <ds:DigestMethod Algorithm...#sha1" />
(M008)      </xenc:EncryptionMethod>
(M009)      <ds:KeyInfo >
(M010)        <wsse:SecurityTokenReference >
(M011)          <wsse:KeyIdentifier EncodingType="...#Base64Binary"
ValueType="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
(M012)        </wsse:SecurityTokenReference>
(M013)      </ds:KeyInfo>
(M014)      <xenc:CipherData>
(M015)        <xenc:CipherValue>TE...=</xenc:CipherValue>
(M016)      </xenc:CipherData>
(M017)    </xenc:EncryptedKey>
(M018)    <wsc:DerivedKeyToken  Algorithm=".../p_sha1"  wsu:Id="DKey1">
(M019)       wsse:SecurityTokenReference  wsse11:TokenType="...#EncryptedKey">
(M020)        <wsse:Reference ValueType="...#EncryptedKey" URI="#encKey"/>
(M021)      </wsse:SecurityTokenReference>
(M022)      <wsc:Generation>0</wsc:Generation>
(M023)      <wsc:Length>32</wsc:Length>
(M024)      <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
(M025)      <wsc:Nonce>39...=</wsc:Nonce>
(M026)    </wsc:DerivedKeyToken>
(M027)    <xenc:ReferenceList>
(M028)      <xenc:DataReference URI="#encBody"/>
(M029)    </xenc:ReferenceList>
(M030)    <wsc:DerivedKeyToken  Algorithm=".../p_sha1" wsu:Id="DKey2">
(M031)      <wsse:SecurityTokenReference   wsse11:TokenType="...#EncryptedKey">
(M032)        <wsse:Reference ValueType="...#EncryptedKey" URI="#encKey"/>
(M033)      </wsse:SecurityTokenReference>
(M034)      <wsc:Generation>0</wsc:Generation>
(M035)      <wsc:Length>32</wsc:Length>
(M036)      <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
(M037)      <wsc:Nonce>...=</wsc:Nonce>
(M038)    </wsc:DerivedKeyToken>
(M039)    <wsu:Timestamp wsu:Id="Timestamp" >
(M040)      <wsu:Created>2007-03-26T23:43:13Z</wsu:Created>
(M041)    </wsu:Timestamp>
(M042)    <ds:Signature >
(M043)      <ds:SignedInfo>
(M044)        ...
(M045)        <ds:Reference URI="#Timestamp">...</ds:Reference>
(M046)        <ds:Reference URI="#Body">...</ds:Reference>
(M047)      </ds:SignedInfo>
(M048)      <ds:SignatureValue>Yu...=</ds:SignatureValue>
(M049)      <ds:KeyInfo>
(M050)        <wsse:SecurityTokenReference >
(M051)          <wsse:Reference URI="#DKey2" ValueType=".../dk"/>
(M052)        </wsse:SecurityTokenReference>
(M053)      </ds:KeyInfo>
(M054)    </ds:Signature>
(M055)   </wsse:Security>
(M056)  </env:Header>
(M057)  <env:Body wsu:Id="Body" >
(M058)   <xenc:EncryptedData Id="encBody" Type="...#Content" MimeType="text/xml"
Encoding="UTF-8" >
(M059)     <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
(M060)     <ds:KeyInfo >
(M061)       <wsse:SecurityTokenReference >
(M062)         <wsse:Reference URI="#DKey1" ValueType=".../dk"/>
(M063)       </wsse:SecurityTokenReference>
(M064)     </ds:KeyInfo>
(M065)     <xenc:CipherData>
(M066)       <xenc:CipherValue>p53...f</xenc:CipherValue>
(M067)     </xenc:CipherData>
(M068)   </xenc:EncryptedData>
(M069)  </env:Body>
```

```
(M070) </env:Envelope>
```

Lines (M005) – (M017) is the EncryptedKey information which will be reference using the WSS1.1 #EncryptedKey SecurityTokenReference function. Lines (M010) – (M012) is the security token reference. Because the ProtectionToken disallowed the token from being inserted into the message, a KeyIdentifier is used instead of a reference to an included token. In addition, Line (M011) the KeyIdentifier has the value type of #ThumbprintSHA1 for Thumbprint Reference, it is required by the policy line (P011) of the Thumbprint Reference.

Lines (M018) – (M022) is the first wsc:DerivedKeyToken. It is derived from the EncryptedKey of lines (M005) – (M017), which is referenced using the WSS1.1 #EncryptedKey SecurityTokenReference mechanism contained in lines (M019) – (M021), and used for body encryption and it is referenced on line (M062).

Lines (M027) – (M029) is an encryption data reference that references the encrypted body of the message on lines (M058) – (M068). The encryption was required by the EncryptedParts assertion of the input message policy.

Lines (M030) – (M038) is the second wsc:DerivedKeyToken. It is derived from the EncryptedKey of lines (M005) – (M017) and used for signature referenced on line (M051).

Lines (M039) – (M041) contain a timestamp for the message as required by the IncludeTimestamp assertion.

Lines (M042) – (M054) contain the message signature.

Line (M045) indicates the message timestamp is included in the signature as required by the IncludeTimestamp assertion definition.

Line (M046) indicates the message body is included in the signature as required by the SignedParts assertion of the input message policy.

2.2.4 (WSS1.1) Mutual Authentication with X.509 Certificates, Sign, Encrypt

This scenario is based on the the “Examples of Secure Web Service Message Exchange Document” [WS-SECURE-INTEROP] (see also sec 2.1.4)

Client and server X509 certificates are used for client and server authorization respectively. Request is signed using K, then encrypted using K, K is ephemeral key protected for server's certificate. Signature corresponding to K is signed using client certificate. Response is signed using K, encrypted using K, encrypted key K is not included in response. Alternatively, derived keys can be used for each of the encryption and signature operations by simply adding an sp:RequireDerivedKeys assertion.

The policy is as follows:

```
(P001) <wsp:Policy wsu:Id="WSS11_AnonymousForX509SignEncrypt_Policy">
(P002)   <wsp:ExactlyOne>
(P003)     <wsp:All>
(P004)       <sp:SymmetricBinding>
(P005)         <wsp:Policy>
(P006)           <sp:ProtectionToken>
(P007)             <wsp:Policy>
(P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
(P009)                 <wsp:Policy>
(P010)                   <sp:RequireDerivedKeys/>
(P011)                   <sp:RequireThumbprintReference/>
(P012)                   <sp:WssX509V3Token11/>
(P013)                 </wsp:Policy>
(P014)               </sp:X509Token>
(P015)             </wsp:Policy>
(P016)           </sp:ProtectionToken>
(P017)           <sp:AlgorithmSuite>
(P018)             <wsp:Policy>
```

```
(P019)               <sp:Basic256/>
(P020)             </wsp:Policy>
(P021)           </sp:AlgorithmSuite>
(P022)           <sp:Layout>
(P023)             <wsp:Policy>
(P024)               <sp:Strict/>
(P025)             </wsp:Policy>
(P026)           </sp:Layout>
(P027)           <sp:IncludeTimestamp/>
(P028)           <sp:OnlySignEntireHeadersAndBody/>
(P029)         </wsp:Policy>
(P030)       </sp:SymmetricBinding>
(P031)       <sp:EndorsingSupportingTokens>
(P032)         <wsp:Policy>
(P033)           <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P034)             <wsp:Policy>
(P035)               <sp:RequireThumbprintReference/>
(P036)               <sp:WssX509V3Token11/>
(P037)             </wsp:Policy>
(P038)           </sp:X509Token>
(P039)         </wsp:Policy>
(P040)       </sp:EndorsingSupportingTokens>
(P041)       <sp:Wss11>
(P042)         <wsp:Policy>
(P043)           <sp:MustSupportRefKeyIdentifier/>
(P044)           <sp:MustSupportRefIssuerSerial/>
(P045)           <sp:MustSupportRefThumbprint/>
(P046)           <sp:MustSupportRefEncryptedKey/>
(P047)           <sp:RequireSignatureConfirmation/>
(P048)         </wsp:Policy>
(P049)       </sp:Wss11>
(P050)     </wsp:All>
(P051)   </wsp:ExactlyOne>
(P052) </wsp:Policy>

(P053) <wsp:Policy wsu:Id=" WSS11_X509DKSignEncrypt_input_policy">
(P054)   <wsp:ExactlyOne>
(P055)     <wsp:All>
(P056)       <sp:SignedParts>
(P057)         <sp:Body/>
(P058)       </sp:SignedParts>
(P059)       <sp:EncryptedParts>
(P060)         <sp:Body/>
(P061)       </sp:EncryptedParts>
(P062)     </wsp:All>
(P063)   </wsp:ExactlyOne>
(P064) </wsp:Policy>

(P065) <wsp:Policy wsu:Id=" WSS11_X509DKSignEncrypt_output_policy">
(P066)   <wsp:ExactlyOne>
(P067)     <wsp:All>
(P068)       <sp:SignedParts>
(P069)         <sp:Body/>
(P070)       </sp:SignedParts>
(P071)       <sp:EncryptedParts>
(P072)         <sp:Body/>
(P073)       </sp:EncryptedParts>
(P074)     </wsp:All>
(P075)   </wsp:ExactlyOne>
(P076) </wsp:Policy>
```

Lines (P004) – (P030) contain the SymmetricBinding assertion which indicates that the derived key token must be used for both message signature and message encryption.

Lines (P007) – (P016) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P014) indicate that the initiator token must be an X.509 token that must not be included with all messages sent between the recipient and Recipient.

Line (P010) dictates the derived key is required. Line (P012) dictates the X.509 token must be an X.509v3 security token as described in the WS-Security 1.1 X.509 Token Profile. According to the MustSupportRefThumbprint assertion on line (P043) and RequireThumbprintReference on line (P011), a Thumbprint Reference of KeyIdentifier must be used to identify the token in any messages where the token is used.

Line (P027) requires the inclusion of a timestamp.

Lines (P031) – (P040) contain the EndorsingSupportingTokens assertion which indicates that the message signature should be endorsed by client's X509 certificate on line (P033) – (P038). Line (P033) IncludeToken=".../AlwaysToRecipient" indicates the endorsing is only required when the message is sent to recipient. Line (P036) dictates the X.509 token must be an X.509v3 security token as described in the WS-Security 1.1 X.509 Token Profile. and RequireThumbprintReference on line (P035), a Thumbprint Reference of KeyIdentifier must be used to identify the token in any messages where the token is used.

Lines (P041) – (P049) contain some WS-Security 1.1 related interoperability requirements, specifically support for key identifier, issuer serial number, thumbprint, encrypted key references, and requires signature confirmation on the response.

Lines (P053) – (P064) contain a policy that is attached to the input message. Lines (P055) – (P058) require that the body of the input message must be signed. Lines (P059) – (P061) require the body of the input message must be encrypted.

Lines (P065) – (P076) contain a policy that is attached to the output message. Lines (P067) – (P070) require that the body of the output message must be signed. Lines (P071) – (P073) require the body of the output message must be encrypted.

An example of an input message that conforms to the above stated policy is as follows.

Note: this message uses WS-SecureConversation as a means to meet the requirements of the policy, however, aside from using the wsc:DerivedKeyToken elements to meet the policy requirements for the RequireDerivedKeys assertion (P010) the general protocol mechanisms described in WS-SecureConversation for SecurityContextTokens are not explicitly demonstrated.

```
(M001) <?xml version="1.0" encoding="utf-8" ?>
(M002) <env:Envelope xmlns:env="..." xmlns:xenc="http..." xmlns:ds="..." xmlns:wsu="...">
(M003)   <env:Header>
(M004)     <wsse:Security xmlns:wsse="..." xmlns:wsse11="..." env:mustUnderstand="1">
(M005)       <xenc:EncryptedKey Id="encKey" >
(M006)         <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
(M007)           <ds:DigestMethod Algorithm="...#sha1" />
(M008)         </xenc:EncryptionMethod>
(M009)         <ds:KeyInfo >
(M010)           <wsse:SecurityTokenReference  >
(M011)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
(M012)                 ValueType="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
(M013)           </wsse:SecurityTokenReference>
(M014)         </ds:KeyInfo>
(M015)         <xenc:CipherData>
(M016)           <xenc:CipherValue>eI...=</xenc:CipherValue>
(M017)         </xenc:CipherData>
(M018)       </xenc:EncryptedKey>
(M019)       <wsse:BinarySecurityToken  ValueType="...#X509v3"
EncodingType="...#Base64Binary">MI...=</wsse:BinarySecurityToken>
(M020)       <wsc:DerivedKeyToken xmlns:wsc=".../sc" Algorithm=".../p_sha1"
wsu:Id="derivedKeyToken1">
(M021)         <wsse:SecurityTokenReference wsse11:TokenType="...#EncryptedKey" >
(M022)           <wsse:Reference ValueType="...#EncryptedKey" URI="#encKey"/>
(M023)         </wsse:SecurityTokenReference>
(M024)         <wsc:Generation>0</wsc:Generation>
(M025)         <wsc:Length>32</wsc:Length>
(M026)         <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
(M027)         <wsc:Nonce>+R...=</wsc:Nonce>
(M028)       </wsc:DerivedKeyToken>
```

```
(M029)        <wsu:Timestamp wsu:Id="Timestamp" >
(M030)          <wsu:Created>2007-03-31T04:27:21Z</wsu:Created>
(M031)        </wsu:Timestamp>
(M032)        <n1:ReferenceList xmlns:n1=".../xmlenc#">
(M033)          <n1:DataReference URI="#encBody"/>
(M034)        </n1:ReferenceList>
(M035)        <wsc:DerivedKeyToken xmlns:wsc=".../sc" Algorithm=".../p_sha1"
wsu:Id="derivedKeyToken2">
(M036)          <wsse:SecurityTokenReference  wsse11:TokenType="...#EncryptedKey" >
(M037)            <wsse:Reference ValueType="...EncryptedKey" URI="#encKey"/>
(M038)          </wsse:SecurityTokenReference>
(M039)          <wsc:Generation>0</wsc:Generation>
(M040)          <wsc:Length>32</wsc:Length>
(M041)          <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
(M042)          <wsc:Nonce>wL...=</wsc:Nonce>
(M043)        </wsc:DerivedKeyToken>
(M044)        <ds:Signature  Id="messageSignature">
(M045)          <ds:SignedInfo>
(M046)            ...
(M047)            <ds:Reference URI="#Timestamp">
(M048)              ...
(M049)            </ds:Reference>
(M050)            <ds:Reference URI="#Body">
(M051)              ...
(M052)            </ds:Reference>
(M053)          </ds:SignedInfo>
(M054)          <ds:SignatureValue>abcdefg=</ds:SignatureValue>
(M055)          <ds:KeyInfo>
(M056)            <wsse:SecurityTokenReference >
(M057)              <wsse:Reference URI="#derivedKeyToken2" ValueType=".../dk"/>
(M058)            </wsse:SecurityTokenReference>
(M059)          </ds:KeyInfo>
(M060)        </ds:Signature>
(M061)        <ds:Signature >
(M062)          <ds:SignedInfo>
(M063)            ...
(M064)            <ds:Reference URI="#messageSignature">
(M065)              ...
(M066)            </ds:Reference>
(M067)          </ds:SignedInfo>
(M068)          <ds:SignatureValue>hijklmnop=</ds:SignatureValue>
(M069)          <ds:KeyInfo>
(M070)            <wsse:SecurityTokenReference >
(M071)              <wsse:KeyIdentifier EncodingType="...#Base64Binary"
ValueType="...#ThumbprintSHA1">Pj...=</wsse:KeyIdentifier>
(M072)            </wsse:SecurityTokenReference>
(M073)          </ds:KeyInfo>
(M074)        </ds:Signature>
(M075)      </wsse:Security>
(M076)    </env:Header>
(M077)    <env:Body wsu:Id="Body" >
(M078)      <xenc:EncryptedData Id="encBody" Type="...#Content" MimeType="text/xml"
Encoding="UTF-8" >
(M079)        <xenc:EncryptionMethod Algorithm="http...#aes256-cbc"/>
(M080)        <ds:KeyInfo >
(M081)          <wsse:SecurityTokenReference >
(M082)            <wsse:Reference URI="#derivedKeyToken1" ValueType=".../dk"/>
(M083)          </wsse:SecurityTokenReference>
(M084)        </ds:KeyInfo>
(M085)        <xenc:CipherData>
(M086)          <xenc:CipherValue>70...=</xenc:CipherValue>
(M087)        </xenc:CipherData>
(M088)      </xenc:EncryptedData>
(M089)    </env:Body>
(M090)  </env:Envelope>
(M091)
```

Lines (M005) – (M017) is the EncryptedKey information which will be reference using the WSS1.1 #EncryptedKey SecurityTokenReference function. Lines (M010) – (M012) is the security token reference.

Lines (M010) – (M012) is the security token reference. Because the ProtectionToken disallowed the token from being inserted into the message, a KeyIdentifier is used instead of a reference to an included token. In addition, Line (M011) the KeyIdentifier has the value type of #ThumbprintSHA1 for Thumbprint Reference, and it is required by the policy line (P011) of the Thumbprint Reference.

Line (M019) is an X509 BinarySecurityToken that contains the actual X509 certificate that is used by the endorsing signature described below. The token is required to be present in the message based on the line (P033) that requires this token for the message sent to the recipient.

Lines (M020) – (M027) is the first derived key token. It is derived from the EncryptedKey of lines (M005) – (M017) and used for body encryption referenced on line (M081).

Lines (M028) – (M030) contain a Timestamp element for the message as required by the IncludeTimestamp assertion.

Lines (M031) – (M033) contain an encryption data reference that references the encrypted body of the message on lines (M077) – (M087). The encryption was required by the EncryptedParts assertion of the input message policy.

Lines (M034) – (M042) is the second derived key token. It is derived from the EncryptedKey of lines (M005) – (M017) and used by the signature that references it on line (M056).

Lines (M043) – (M059) contain the message signature. Lines (M054) – (M058) is its KeyInfo block that indicates the second derived key token should be used to verify the message signature.

Line (M057) indicates the message timestamp is included in the signature as required by the IncludeTimestamp assertion definition.

Line (M060) indicates the message body is included in the signature as required by the SignedParts assertion of the input message policy.

Lines (M060) – (M073) contain the endorsing signature. It signs the message signature, referenced on line (M063), per EndorsingSupportingTokens assertion policy requirement on lines (P031) – (P040). Line (M070), the KeyIdentifier, has the value type of #ThumbprintSHA1 for Thumbprint Reference, and it is required by the policy line (P035) of the Thumbprint Reference. This Thumbprint Reference must match the Thumbprint of the X509 Certificate contained in the BinarySecurityToken on line (M019), based on the IncludeToken requirement line (P033).

An example of an output message that conforms to the above stated policy follows:

```
(R001)     <env:Envelope xmlns:env="..." xmlns:wsu="...">
(R002)       <env:Header>
(R003)         <wsse:Security env:mustUnderstand="1" xmlns:wsse="...">
(R004)           <wsu:Timestamp wsu:Id="Timestamp" >
(R005)             <wsu:Created>2007-03-31T04:27:25Z</wsu:Created>
(R006)           </wsu:Timestamp>
(R007)           <wsc:DerivedKeyToken wsu:Id="derivedKeyToken1" xmlns:wsc=".../sc">
(R008)             <wsse:SecurityTokenReference>
(R009)               <wsse:KeyIdentifier ValueType="...1.1#EncryptedKeySHA1"
(R010)                >nazB6DwNC9tcwFsghoSYWXLf2wk=</wsse:KeyIdentifier>
(R011)             </wsse:SecurityTokenReference>
(R012)             <wsc:Offset>0</wsc:Offset>
(R013)             <wsc:Length>24</wsc:Length>
(R014)             <wsc:Nonce>NfSNXZLYAA8mocQz19KWjg==</wsc:Nonce>
(R015)           </wsc:DerivedKeyToken>
(R016)           <wsc:DerivedKeyToken wsu:Id="derivedKeyToken2" xmlns:wsc=".../sc">
(R017)             <wsse:SecurityTokenReference>
(R018)               <wsse:KeyIdentifier ValueType="...1.1#EncryptedKeySHA1"
(R019)                >nazB6DwNC9tcwFsghoSYWXLf2wk=</wsse:KeyIdentifier>
(R020)             </wsse:SecurityTokenReference>
(R021)             <wsc:Nonce>lF/CtyQ9d1Ro8E3+uZYmgQ==</wsc:Nonce>
(R022)           </wsc:DerivedKeyToken>
(R023)           <enc:ReferenceList xmlns:enc=".../xmlenc#">
(R024)             <enc:DataReference URI="#encBody"/>
(R025)           </enc:ReferenceList>
(R026)           <wsse11:SignatureConfirmation wsu:Id="sigconf1"
(R027)               Value="abcdefg=" xmlns:wsse11="..."/>
```

```
(R028)          <wsse11:SignatureConfirmation wsu:Id="sigconf2"
(R029)              Value="hijklmnop=" xmlns:wsse11="..."/>
(R030)          <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
(R031)            <SignedInfo>
(R032)              <CanonicalizationMethod
(R033)                  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(R034)              <SignatureMethod
(R035)                  Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
(R036)              <Reference URI="#msgBody">
(R037)                ...
(R038)              </Reference>
(R039)              <Reference URI="#Timestamp">
(R040)                ...
(R041)              </Reference>
(R042)              <Reference URI="#sigconf1">
(R043)                ...
(R044)              </Reference>
(R045)              <Reference URI="#sigconf2">
(R046)                ...
(R047)              </Reference>
(R048)            </SignedInfo>
(R049)            <SignatureValue>3rAxsfJ2LjF7liRQX2EH/0DBmzE=</SignatureValue>
(R050)            <KeyInfo>
(R051)              <wsse:SecurityTokenReference>
(R052)                <wsse:Reference URI="#derivedKeyToken1"/>
(R053)              </wsse:SecurityTokenReference>
(R054)            </KeyInfo>
(R055)          </Signature>
(R056)        </wsse:Security>
(R057)      </env:Header>
(R058)      <env:Body wsu:Id="msgBody">
(R059)        <enc:EncryptedData Id="encBody" Type="...xmlenc#Content"
(R060)            xmlns:enc=".../xmlenc#">
(R061)          <enc:EncryptionMethod Algorithm=".../xmlenc#aes256-cbc"/>
(R062)          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
(R063)            <wsse:SecurityTokenReference xmlns:wsse="...">
(R064)              <wsse:Reference URI="#derivedKeyToken2"/>
(R065)            </wsse:SecurityTokenReference>
(R066)          </KeyInfo>
(R067)          <enc:CipherData>
(R068)            <enc:CipherValue>y+eV...pu</enc:CipherValue>
(R069)          </enc:CipherData>
(R070)        </enc:EncryptedData>
(R071)      </env:Body>
(R072)    </env:Envelope>
```

Lines (R001)-(R072) contain the Response message returned to the Initiator.

Lines (R004)-(R006) contain the Timestamp required by the Policy (P027).

Lines (R007)-(R015) and (R016)-(R022) contain the information necessary for the Initiator to derive the keys using the WS-SecureConversation shared secret, the identified key (R010) for DK1 and (R019) for DK2, which reference the same key, and the Nonce (R014) for DK1 and (R021) for DK2, which are different for the two derived keys.

Lines (R023)-(R025) contain a ReferenceList that indicates the message Body (R058) contains the data to be encrypted. The encryption was required by the output policy (P079).

Lines (R026)-(R027) contain the SignatureConfirmation for the SignatureValue in the request message (M054) which was required to be included by the policy (P047) RequireSignatureConfirmation.

Lines (R028)-(R029) contain the SignatureConfirmation for the 2nd SignatureValue in the request message (M068), which is also required by the policy (P047).

Lines (R030)-(R055) contain the response message signature that covers the Timestamp element (R039)->(R004) required to be signed by the policy (P027), the two SignatureConfirmation elements (R042)->(R026) and (R045)->(R028), which are required to be signed because they are required elements of the policy (P047), and the message Body (R036)->(R058), which is required to be signed by the output message policy (P076). The signature may be verified using derivedKeyToken1 as indicated on line (R052).

Lines (R059)-(R070) contain the encrypted data as required by the policy (P079) and may be decrypted using derivedKeyToken2 as indicated on line (R064).

2.3 SAML Token Authentication Scenario Assertions

For SAML, the combination of SAML and WSS version numbers is supported (WssSamlV11Token10, WssSamlV11Token11, WssSamlV20Token11).

Instead of explicitly including the SAML Assertion, a wsse:KeyIdentifier reference can also be used. To enable this last behavior, the IncludeToken attribute is set to http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never.

In all of the SAML scenarios, the SAML Assertion ConfirmationMethod expected to be used by the Initiator is communicated implicitly by the context of the sp: security binding in use and type of sp: element containing the SamlToken. There are 3 general SAML ConfirmationMethod use cases covered: holder-of-key (hk), sender-vouches (sv), and bearer (br).

For the **holder-of-key** case, there is always a contained reference (or value) in the SAML Assertion to key material that may be used for message protection in the scenario. The hk case can be assumed if the WssSamlV**Token** element appears either in the sp:InitiatorToken element of the sp:AsymmetricBinding element, or in the sp:ProtectionToken element of the sp:SymmetricBinding element. In the sp:TransportBinding case, if the WssSamlV**Token** element appears in an sp:EndorsingSupportingTokens or sp:SignedEndorsingSupportingTokens element, the SAML Assertion type can also be assumed to be hk.

The **sender-vouches** case can be assumed if the WssSamlV**Token** version will always appear in an sp:SignedSupportingTokens element to indicate that the SAML Authority associated with this token is the Initiator that signs the message.

The **bearer** case can be assumed if the WssSamlV**Token** version appears in an sp:SupportingTokens element (it may be signed as a SignedPart, but it is not involved in the message protection).

It is recognized that other uses cases might exist, where these guidelines might need further elaboration in order to address all contingencies, however that is outside the scope of the current version of this document.

Note: in the discussions below the term “SamlToken” or “SamlToken assertion” refers to the policy requirement that a “SAML Assertion” be used as that token.

Note: SAML Assertions have issuers. In addition, these Assertions are generally signed with an X509 certificate. In general, the SAML IssuingAuthority may be regarded as the Subject of the X509 certificate, which is trusted by a RelyingParty. In general, as well, there is usually a known mapping between the Issuer identified within the SAML Assertion and the Subject of the X509 certificate used to sign that Assertion. It will be assumed in this document that the SAML IssuingAuthority may be identified by either of these identities and that, in general, a RelyingParty will check the details as necessary.

The concept of the sp:”message signature” within the context of the **sp:TransportBinding** is not clearly identified within the current version of [WS-SecurityPolicy], however, there are certain inferences that are used in the use cases that follow that are identified here for reference.

- Based on the diagrams in section 8 of [WS-SecurityPolicy], which show messages with a “message signature” followed by a diagram showing use of transport security the only difference is that the message signature diagrams contain a block labelled “Sig1”, which is the message signature, and the transport security diagrams do not have this block.
- Considering the fact that when [SSL] Client Certificates are used for client authentication, that the client uses the client certificate private key to sign hash of SSL handshake messages in an SSL CertificateVerify message that is part of the SSL protocol, the assumption is made that subsequent data sent by the client on this link is effectively protected for the purposes of data

integrity and confidentiality, and that the data sent on the link is authorized to be sent by the holder of the private key associated with the client certificate.

- Based on the considerations in the bullets above, and with intention of maintaining functional consistency with the WS-SecurityPolicy notions of sp: message signature, sp:SignedSupportingTokens, and sp:SignedEndorsingSupportingTokens, use of client certicates with SSL will be considered equivalent to having a “virtual message signature” provided by the Initiator’s client certificate which covers all the data in the SOAP request that the Initiator sends on the SSL link.
- As such, in the above context, the client certificate may be regarded as providing both a virtual Signing function for tokens and Timestamp that appear in the wsse:Security header, as well as a virtual Endorsing function for tokens that contain the client certificate or a reference to the client certificate that appear in the wsse:Security header.

The net effect of the above assumptions for the SAML use cases in this section that use the **sp:TransportBinding** is that if a **client certifcate is required** to be used with the **sp:HttpsToken** assertion then:

1. A SAML **sender-vouches** Assertion contained in a wsse:Security header block may be considered to be an **sp:SignedSupportingToken** when used in an sp:TransportBinding using an sp:HttpsToken assertion that contains an sp:RequireClientCertificate assertion, because the client certificate is being used as the IssuingAuthority behind the SAML sender-vouches Assertion, which requires the IssuingAuthority to sign the combined message and Assertion.
2. A SAML **holder-of-key** Assertion contained in a wsse:Security header block may be considered to be an **sp:SignedEndorsingSupportingToken** when used in an sp:TransportBinding using an sp:HttpsToken assertion that contains an sp:RequireClientCerfticate assertion AND that the either the client certificate or a reference to it is contained in the saml:SubjectConfirmationData/ds:KeyInfo element of the SAML holder-of-key Assertion.
3. A SAML **bearer** Assertion contained in a wsse:Security header block may only be considered to be an **sp:SupportingToken**, because they are only “incidentally” covered by the virtual message signature as a “pass through” token provided by the Requestor to be evaluated by the Recipient that is being “passed through” by the Initiator, but which the Initiator takes no responsibility.

Ultimately, the responsibilities associated with the granting access to resources is determined by the agreements between RelyingParties and IssuingAuthorities. Those agreements need to take into consideration the security characteristics of the bindings which support access requests as to what kind of bindings are required to “adequately protect” the requests for the associated business purposes. These examples are intended to show how SAML Assertions can be used in a variety of WS-SecurityPolicy contexts and how the different SAML token types (hk, sv, bearer) may be used in different configurations relating the IssuingAuthority, the Requestor, the Initiator, the Recipient, and the RelyingParty.

2.3.1 WSS 1.0 SAML Token Scenarios

2.3.1.1 (WSS1.0) SAML1.1 Assertion (Bearer)

Initiator adds a SAML assertion (bearer) representing the Requestor to the SOAP security header.

Since the SamlToken is listed in the SupportingTokens element, it will not explicitly be covered by a message signature. The Initiator may infer that a Saml Bearer Assertion is acceptable to meet this requirement, because the Initiator is not required to explicitly cover a SupportingToken with a signature.

The SAML assertion itself could be signed. The IssuingAuthority in this scenario is the Issuer (and signer, if the Assertion is signed) of the SAML Assertion. The Initiator simply passes the token through and is not actively involved in the trust relationship between the IssuingAuthority that issued the SAML Assertion and the Requestor who is the Subject of the SAML Assertion.

This scenario might be used in a SAML Federation application where a browser-based user with a SAML Assertion indicating that user's SSO (Single Sign On) credential has submitted a request to a portal using the Assertion as a credential (either directly or indirectly via a SAML Artifact [SAML11]), and the portal as Initiator is generating a web service request on behalf of this user, but the trust that the Recipient has for the Requestor is based on the Assertion and its IssuingAuthority, not the Initiator who delivered the request.

```
(P001) <wsp:Policy>
(P002)   <sp:SupportingTokens>
(P003)     <wsp:Policy>
(P004)       <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-
     sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P005)         <wsp:Policy>
(P006)           <sp:WssSamlV11Token10/>
(P007)         </wsp:Policy>
(P008)       </sp:SamlToken>
(P009)     </wsp:Policy>
(P010)   </sp:SupportingTokens>
(P011) </wsp:Policy>
```

Lines (P001)-(P011) contain the endpoint wsp:Policy, which contains no bindings because none is required to access the service protected by this policy, however the service does require that the Requestor provide a Supporting Token in the sp:SupportingTokens element (P002)-(P010), which must be an sp:SamlToken (P004)-(P008), which must be an sp:WssSamlV11Token10 (P006), which means that the SamlToken must be a SAML 1.1 saml:Assertion that is sent in compliance with the [WSS10-SAML11-PROFILE].

As explained in section 2.3 above, the fact that the sp:SamlToken is simply in an sp:SupportingTokens element indicates to the Initiator that a SAML bearer Assertion is what is expected to accompany the request.

The following is a sample request taken from [WSS11-SAML1120-PROFILE] that complies with the WSS 1.0 compatible policy above

```
(M001)   <S12:Envelope xmlns:S12="...">
(M002)     <S12:Header>
(M003)       <wsse:Security xmlns:wsse="...">
(M004)         <saml:Assertion xmlns:saml="..."
(M005)             AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
(M006)             IssueInstant="2003-04-17T00:46:02Z"
(M007)             Issuer="www.opensaml.org"
(M008)             MajorVersion="1"
(M009)             MinorVersion="1">
(M010)           <saml:AuthenticationStatement>
(M011)             <saml:Subject>
(M012)               <saml:NameIdentifier
(M013)                   NameQualifier="www.example.com"
(M014)    Format="urn:oasis:names:tc:SAML:1.1:nameidformat:X509SubjectName">
(M015)                 uid=joe,ou=people,ou=saml-demo,o=baltimore.com
(M016)               </saml:NameIdentifier>
(M017)               <saml:SubjectConfirmation>
(M018)                 <saml:ConfirmationMethod>
(M019)                   urn:oasis:names:tc:SAML:1.0:cm:bearer
(M020)                 </saml:ConfirmationMethod>
(M021)               </saml:SubjectConfirmation>
(M022)             </saml:Subject>
(M023)           </saml:AuthenticationStatement>
(M024)         </saml:Assertion>
(M025)       </wsse:Security>
(M026)     </S12:Header>
(M027)     <S12:Body>
```

```
(M028)        . . .
(M029)      </S12:Body>
(M030)    </S12:Envelope>
```

Lines (M001)-(M030) contains the SOAP:Envelope, which contains the SOAP:Header (M002)-(M026) and the SOAP:Body (M027)-(M029).

The SOAP Header contains a wsse:Security header block (M003)-(M025) which simply contains the saml:Assertion.

The saml:Assertion (M004)-(M024) is Version 1.1 (M008)-(M009), which is required by the policy sp:WssSamlV11Token10 assertion (P006). The Assertion contains a saml:AuthenticationStatement (M010)-(M023) that contains a saml:NameIdentifier (M012)-(M016) that identifies the saml:Subject, who is the Requestor (who submitted the request from a browser) and it contains a saml:SubjectConfirmation element (M017)-(M021), which specifies the saml:ConfirmationMethod to be “bearer” (M019), which meets the policy requirement (P006).

For general context to relate this scenario to Figure 1, the Requestor at a browser will have obtained either the saml:Assertion above or an Artifact identifying that Assertion from an IssuingAuthority and sent it to the portal in the context of some request the Requestor is trying to make. The portal recognizes that to meet the needs of this request that it must invoke a web service that has publicized the policy above (P001)-(P011). Therefore, the portal will now take on the role of Initiator (Fig 1) and assemble a SOAP request (M001)-(M030) and submit it to the service as described in detail above.

2.3.1.2 (WSS1.0) SAML1.1 Assertion (Sender Vouches) over SSL

This scenario is based on first WSS SAML Profile InterOp [WSS10-SAML11-INTEROP Scenario #2 section 4.4.4]

Initiator adds a SAML Assertion (sv) to the SOAP Security Header. Because the TransportBinding requires a Client Certificate AND the SAML token is in a SignedSupportingTokens element, when the Initiator uses the Client Certificate to protect the message under SSL, the Initiator may be considered as effectively “signing” the SAML sv Assertion and acting as a SAML Authority (i.e. the issuer of the sv Assertion). By including the sv assertion in the header and using the Client Certificate to protect the message, the Initiator takes responsibility for binding the Requestor, who is the Subject of the Assertion to the contents of the message.

> Note: because SSL does not retain cryptographic protection of the message after the message is delivered, messages protected using only this mechanism cannot be used as the basis for non-repudiation.

```
(P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
wsu:Id="Wss10SamlSvV11Tran_policy">
(P002)   <sp:TransportBinding>
(P003)     <wsp:Policy>
(P004)       <sp:TransportToken>
(P005)         <wsp:Policy>
(P006)           <sp:HttpsToken>
(P007)             <wsp:Policy>
(P008)               <sp:RequireClientCertificate>
(P009)             </wsp:Policy>
(P010)           </sp:HttpsToken>
(P011)         </wsp:Policy>
(P012)       </sp:TransportToken>
(P013)      <sp:AlgorithmSuite>
(P014)         <wsp:Policy>
(P015)           <sp:Basic256 />
(P016)         </wsp:Policy>
(P017)       </sp:AlgorithmSuite>
(P018)       <sp:Layout>
(P019)         <wsp:Policy>
(P020)           <sp:Strict />
```

```
(P021)          </wsp:Policy>
(P022)        </sp:Layout>
(P023)        <sp:IncludeTimestamp />
(P024)      </wsp:Policy>
(P025)    </sp:TransportBinding>
(P026)    <sp:SignedSupportingTokens>
(P027)      <wsp:Policy>
(P028)        <sp:SamlToken sp:IncludeToken=”http://docs.oasis-open.org/ws-
sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient”>
(P029)          <wsp:Policy>
(P030)            <sp:WssSamlV11Token10/>
(P031)          </wsp:Policy>
(P032)        </sp:SamlToken>
(P033)      </wsp:Policy>
(P034)    </sp:SignedSupportingTokens>
(P035) </wsp:Policy>
```

Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected by a secure transport protocol such as SSL or TLS.

Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the transport token using the Basic256 algorithm suite (P015).

In addition, because this is SAML sender-vouches, a client certificate is required (P008) as the basis of trust for the SAML Assertion and for the content of the message [WSS10-SAML11-INTEROP section 4.3.1].

The layout requirement in this case (P018)-(P022) is automatically met (or may be considered moot) since there are no cryptographic tokens required to be present in the WS-Security header.

A timestamp (P023) is required to be included in an acceptable message.

Lines (P026)-(P034) contain a SignedSupportingTokens assertion, which indicates that the referenced token is effectively “signed” by the combination of usage of the client certificate for SSL authentication and the cryptographic protection of SSL. However, the “signed” characteristic will not be present after the message is delivered from the transport layer to the recipient.

Lines (P028)-(P032) indicate the signed token is a SAML Assertion (sender-vouches as described above in section 2.3) and on line (P030) that it is a SAML 1.1 Assertion and that the WS-Security 1.0 SAML Profile [WSS10-SAML11-PROFILE] is being used.

This scenario is based on first WSS SAML Profile InterOp [WSS10-SAML11-INTEROP “Scenario #2 - Sender-Vouches: Unsigned: SSL” section 4.4.4]

Here is the example request from that scenario:

```
(M001)    <?xml version="1.0" encoding="utf-8" ?>
(M002)    <soap:Envelope
(M003)      xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
(M004)      xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
(M005)      xmlns:xsd=http://www.w3.org/2001/XMLSchema
(M006)      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401
(M007)    -wss-wssecurity-secext-1.0.xsd"
(M008)      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss
(M009)    -wssecurity-utility-1.0.xsd">
(M010)     <soap:Header>
(M011)      <wsse:Security soap:mustUnderstand="1">
(M012)          <wsu:Timestamp>
(M013)            <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
(M014)          </wsu:Timestamp>
(M015)          <saml:Assertion
(M016)              xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
(M017)              MajorVersion="1" MinorVersion="1"
(M018)              AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
(M019)              Issuer=www.opensaml.org
```

```
(M020)             IssueInstant="2002-06-19T16:58:33.173Z">
(M021)           <saml:Conditions
(M022)               NotBefore="2002-06-19T16:53:33.173Z"
(M023)               NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
(M024)           <saml:AuthenticationStatement
(M025)               AuthenticationMethod=
(M026)                 "urn:oasis:names:tc:SAML:1.0:am:password"
(M027)               AuthenticationInstant="2002-06-19T16:57:30.000Z">
(M028)             <saml:Subject>
(M029)               <saml:NameIdentifier
(M030)                   NameQualifier=www.example.com
(M031)                   Format="">
(M032)                 uid=joe,ou=people,ou=saml-demo,o=example.com
(M033)               </saml:NameIdentifier>
(M034)               <saml:SubjectConfirmation>
(M035)                 <saml:ConfirmationMethod>
(M036)                   urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
(M037)                 </saml:ConfirmationMethod>
(M038)               </saml:SubjectConfirmation>
(M039)             </saml:Subject>
(M040)           </saml:AuthenticationStatement>
(M041)         </saml:Assertion>
(M042)     </wsse:Security>
(M043)    </soap:Header>
(M044)    <soap:Body>
(M045)     <Ping xmlns="http://xmlsoap.org/Ping">
(M046)      <text>EchoString</text>
(M047)     </Ping>
(M048)    </soap:Body>
(M049)   </soap:Envelope>
```

Lines (M011)-(M042) contain the WS-Security 1.0 header required by the WssSamlV11Token10 assertion (P030).

Lines (M012)-(M014) contain the wsu:Timestamp required by IncludeTimestamp assertion (P023).

Lines (M015)-(M041) contain the SAML 1.1 Assertion required by the WssSamlV11Token10 assertion (P030).

Note that the additional requirements identified in the policy above are met by the SSL transport capabilities and do not appear in any form in the SOAP message (M001)-(M049).

2.3.1.3 (WSS1.0) SAML1.1 Assertion (HK) over SSL

Initiator adds a SAML assertion (hk) to the SOAP Security Header. Because the TransportBinding requires a Client Certificate AND the SAML token is in an EndorsingSupportingTokens element, the Initiator may be considered to be authorized by the issuer of the hk SAML assertion to bind message content to the Subject of the assertion. If the Client Certificate matches the certificate identified in the hk assertion, the Initiator may be regarded as executing SAML hk responsibility of binding the Requestor, who would be the Subject of the hk assertion, to the content of the message.

In this scenario, the IssuingAuthority is the issuer(signer) of the hk SAML Assertion. The Requestor is the Subject of the Assertion and the Initiator is authorized by the Authority to bind the Assertion to the message using the ClientCertificate identified in the SAML Assertion, which should also be used to sign the timestamp of the message with a separate Signature included in the WS-Security header.

```
(P001)      <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
(P002)          wsu:Id="Wss10SamlHkV11Tran_policy">>
(P003)        <sp:TransportBinding>
(P004)          <wsp:Policy>
(P005)            <sp:TransportToken>
(P006)              <wsp:Policy>
(P007)                <sp:HttpsToken>
```

```
(P008)                  <wsp:Policy>
(P009)                    <sp:RequireClientCertificate>
(P010)                  </wsp:Policy>
(P011)                </sp:HttpsToken>
(P012)              </wsp:Policy>
(P013)            </sp:TransportToken>
(P014)            <sp:AlgorithmSuite>
(P015)              <wsp:Policy>
(P016)                <sp:Basic256 />
(P017)              </wsp:Policy>
(P018)            </sp:AlgorithmSuite>
(P019)            <sp:Layout>
(P020)              <wsp:Policy>
(P021)                <sp:Strict />
(P022)              </wsp:Policy>
(P023)            </sp:Layout>
(P024)            <sp:IncludeTimestamp />
(P025)          </wsp:Policy>
(P026)        </sp:TransportBinding>
(P027)        <sp:SignedEndorsingSupportingTokens>
(P028)          <wsp:Policy>
(P029)            <sp:SamlToken sp:IncludeToken="http://docs.oasis-
            open.org/ws-sx/ws-
            securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P030)              <wsp:Policy>
(P031)                <sp:WssSamlV11Token10/>
(P032)              </wsp:Policy>
(P033)            </sp:SamlToken>
(P034)          </wsp:Policy>
(P035)        </sp:SignedEndorsingSupportingTokens>
(P036)      <wsp:Policy>
```

Section 2.3.2.3 contains an example message that is similar to one that could be used for the policy above, except the Signed Endorsing Supporting Token there is a SAML Version 2.0 token, and a SAML Version 1.1 token would be required here.

2.3.1.4 (WSS1.0) SAML1.1 Sender Vouches with X.509 Certificates, Sign, Optional Encrypt

This scenario is based on the first WSS SAML Profile InterOp [WSS10-SAML11-INTEROP Scenario #3].

In this case, the SAML token is included as part of the message signature and sent only to the Recipient. The message security is provided using X.509 certificates with both requestor and service having exchanged these credentials via some out of band mechanism, which provides the basis of trust of each others' certificates.

In this scenario the SAML Authority is the Initiator who uses the message signature to both provide the integrity of the message and to establish the Initiator as the SAML Authority based on the X509 certificate used to sign the message and the SignedSupportingTokens SAML Assertion. Effectively, the SAML Assertion is being "issued" as part of the message construction process. The Requestor is the Subject of the SAML Assertion. The Initiator knows that the Recipient is expecting it to be the SAML Authority by the fact that the policy specifies that the message requires a SignedSupportingTokens SamlToken.

```
(P001)    <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
            wsu:Id="Wss10SamlSvV11Asymm_endpoint_policy">
(P003)      <wsp:ExactlyOne>
(P004)        <wsp:All>
(P005)          <sp:AsymmetricBinding>
(P006)            <wsp:Policy>
(P007)              <sp:InitiatorToken>
(P008)                <wsp:Policy>
```

```
(P009)                 <sp:X509Token sp:IncludeToken="http://docs.oasis-
         open.org/ws-sx/ws-
         securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P010)                   <wsp:Policy>
(P011)                     <sp:WssX509V3Token10/>
(P012)                   </wsp:Policy>
(P013)                 </sp:X509Token>
(P014)               </wsp:Policy>
(P015)             </sp:InitiatorToken>
(P016)             <sp:RecipientToken>
(P017)               <wsp:Policy>
(P018)                 <sp:X509Token sp:IncludeToken="http://docs.oasis-
         open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
(P019)                   <wsp:Policy>
(P020)                     <sp:WssX509V3Token10/>
(P021)                   </wsp:Policy>
(P022)                 </sp:X509Token>
(P023)               </wsp:Policy>
(P024)             </sp:RecipientToken>
(P025)             <sp:AlgorithmSuite>
(P026)               <wsp:Policy>
(P027)                 <sp:Basic256/>
(P028)               </wsp:Policy>
(P029)             </sp:AlgorithmSuite>
(P030)             <sp:Layout>
(P031)               <wsp:Policy>
(P032)                 <sp:Strict/>
(P033)               </wsp:Policy>
(P034)             </sp:Layout>
(P035)             <sp:IncludeTimestamp/>
(P036)             <sp:OnlySignEntireHeadersAndBody/>
(P037)           </wsp:Policy>
(P038)         </sp:AsymmetricBinding>
(P039)         <sp:Wss10>
(P040)           <wsp:Policy>
(P041)             <sp:MustSupportRefKeyIdentifier/>
(P042)           </wsp:Policy>
(P043)         </sp:Wss10>
(P044)         <sp:SignedSupportingTokens>
(P045)           <wsp:Policy>
(P046)             <sp:SamlToken sp:IncludeToken="http://docs.oasis-
         open.org/ws-sx/ws-
         securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P047)               <wsp:Policy>
(P048)                 <sp:WssSamlV11Token10/>
(P049)               </wsp:Policy>
(P050)             </sp:SamlToken>
(P051)           </wsp:Policy>
(P052)         </sp:SignedSupportingTokens>
(P053)       </wsp:All>
(P054)     </wsp:ExactlyOne>
(P055)   </wsp:Policy>
(P056)
(P057)   <wsp:Policy wsu:Id="Wss10SamlSvV11Asymm_input_policy">
(P058)     <wsp:ExactlyOne>
(P059)       <wsp:All>
(P060)         <sp:SignedParts>
(P061)           <sp:Body/>
(P062)         </sp:SignedParts>
(P063)         <sp:EncryptedParts wsp:Optional="true">
(P064)           <sp:Body/>
(P065)         </sp:EncryptedParts>
(P066)       </wsp:All>
(P067)     </wsp:ExactlyOne>
```

```
(P068)   </wsp:Policy>

(P069)   <wsp:Policy wsu:Id="Wss10SamlSvV11Asymm_output_policy">
(P070)     <wsp:ExactlyOne>
(P071)       <wsp:All>
(P072)         <sp:SignedParts>
(P073)           <sp:Body/>
(P074)         </sp:SignedParts>
(P075)         <sp:EncryptedParts>
(P076)           <sp:Body/>
(P077)         </sp:EncryptedParts>
(P078)       </wsp:All>
(P079)     </wsp:ExactlyOne>
(P080)   </wsp:Policy>
```

Line (P004) is a wsp:All element whose scope carries down to line (P053), which means all 3 of the contained assertions: (P005)-(P038) AsymmetricBinding, (P039)-(P043) WSS10, and (P044)-(P052) SignedSupportingTokens must be complied with by message exchanges to a Web Service covered by this policy.

Lines (P005)-(P038) contain an AsymmetricBinding assertion which indicates that the Initiator's token must be used for the message signature and that the recipient's token must be used for message encryption.

Lines (P007)-(P015) contain the InitiatorToken assertion. Within the InitiatorToken assertion, lines (P009)-(P013) indicate that the Initiator's token must be an X509Token that must always be included as part of the request message (as opposed to an external reference). Line (P011) indicates that the X509 token must be an X.509 V3 signature-verification certificate and used in the manner described in [WSS10-X509-PROFILE].

Lines (P016)-(P023) contain the RecipientToken assertion. Again, this an X.509 V3 certificate (P020) that must be used as described in [WSS10-X509-PROFILE], however the token itself, must never be included in messages in either direction (P018).

Instead (momentarily skipping slightly ahead), policy lines (P039)-(P043), the WSS10 assertion, indicate that the Recipient's token must be referenced by a KeyIdentifier element, which is described in the WS-Security 1.0 core specification [WSS10-SOAPMSG].

Lines (P025)-(P029) contain the AlgorithmSuite assertion, which specifies the sp:Basic256 set of cryptographic components. The relevant values for this example within the sp:Basic256 components are the asymmetric signing algorithm, ds:rsa-sha1, and the digest algorithm, ds:sha1, where "ds:" = "http://www.w3.org/2000/09/xmldsig# " [XML-DSIG].

Lines (P030)-(P034) contain the sp:Layout assertion, which is set to sp:Strict (P032), which governs the required relative layout of various Timestamp, Signature and Token elements in the messages.

Line (P035) IncludeTimestamp means a Timestamp element must be included in the WS-Security header block and signed by the message signature for messages in both directions.

Line (P036) OnlySignEntireHeaderAndBody indicates that the message signature must explicitly cover the SOAP:Body element (not children), and if there are any signed elements in the SOAP:Header they must be direct child elements of the SOAP:Header. However, the one exception where the latter condition is not applied is that if the child of the SOAP:Header is a WS-Security header (i.e. a wsse:Security element), then individual direct child only elements of that Security element may also be signed.

Lines (P044)-(P052) contain a SignedSupportingTokens assertion, which contains only one token (P046)-(P050), a SamlToken, which must always be sent to the Recipient (P046) and it must be a SAML 1.1 Assertion token. Because the SamlToken is in a "SignedSupportingTokens" element, it is implicitly a SAML sender-vouches ConfirmationMethod token as described in section 2.3 above.

Lines (P057)-(P068) contain a policy that applies only to input messages from the Initiator to the Recipient.

Lines (P060)-(P062) specify that the SOAP:Body element of the input message must be signed by the Initiator's message signature.

Lines (P063)-(P065) specify that the SOAP:Body element of the input message may optionally be encrypted (signified by wsp:Optional="true") using the Recipient's encryption token (in this case (P020), it is an X.509 certificate).

Lines (P069)-(P080) contain a policy that applies only to output messages from the Recipient to the Initiator.

Lines (P072)-(P074) specify that the SOAP:Body element of the output message must be signed by the Recipient's message signature.

Lines (P075)-(P077) specify that the SOAP:Body element of the output message must be encrypted by the Initiator's encryption token (in this case (P012), it is also an X.509 certificate).

Here is an example request:

```
(M001)   <?xml version="1.0" encoding="utf-8" ?>
(M002)   <S12:Envelope
(M003)     xmlns:S12=http://schemas.xmlsoap.org/soap/envelope/
(M004)     xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
(M005)     xmlns:xsd=http://www.w3.org/2001/XMLSchema
(M006)     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
(M007)   200401-wss-wssecurity-secext-1.0.xsd"
(M008)     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
(M009)   200401-wss-wssecurity-utility-1.0.xsd"
(M010)     xmlns:ds=http://www.w3.org/2000/09/xmldsig#
(M011)     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
(M012)    <S12:Header>
(M013)     <wsse:Security S12:mustUnderstand="1">
(M014)      <wsu:Timestamp wsu:Id="timestamp">
(M015)        <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
(M016)      </wsu:Timestamp>
(M017)      <saml:Assertion
(M018)        AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
(M019)        IssueInstant="2003-04-17T00:46:02Z"
(M020)        Issuer=www.opensaml.org
(M021)        MajorVersion="1"
(M022)        MinorVersion="1">
(M023)       <saml:Conditions
(M024)         NotBefore="2002-06-19T16:53:33.173Z"
(M025)         NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
(M026)       <saml:AttributeStatement>
(M027)        <saml:Subject>
(M028)         <saml:NameIdentifier
(M029)           NameQualifier=www.example.com
(M030)           Format="">
(M031)          uid=joe,ou=people,ou=saml-demo,o=example.com
(M032)         </saml:NameIdentifier>
(M033)         <saml:SubjectConfirmation>
(M034)          <saml:ConfirmationMethod>
(M035)           urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
(M036)          </saml:ConfirmationMethod>
(M037)         </saml:SubjectConfirmation>
(M038)        </saml:Subject>
(M039)        <saml:Attribute>
(M040)         ...
(M041)        </saml:Attribute>
(M042)        ...
(M043)       </saml:AttributeStatement>
(M044)      </saml:Assertion>
(M045)      <wsse:SecurityTokenReference wsu:id="STR1">
(M046)       <wsse:KeyIdentifier wsu:id="..."
(M047)         ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-
         2004XXwss-saml-token-profile-1.0#SAMLAssertionID">
(M048)         _a75adf55-01d7-40cc-929f-dbd8372ebdfc
(M049)       </wsse:KeyIdentifier>
```

```
(M050)        </wsse:SecurityTokenReference>
(M051)        <wsse:BinarySecurityToken
(M052)          wsu:Id="attesterCert"
(M053)          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
(M054)     200401-wss-x509-token-profile-1.0#X509v3"
(M055)          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
(M056)     200401-wss-soap-message-security-1.0#Base64Binary">
(M057)         MIIEZzCCA9CgAwIBAgIQEmtJZc0...
(M058)        </wsse:BinarySecurityToken>
(M059)        <ds:Signature>
(M060)         <ds:SignedInfo>
(M061)          <ds:CanonicalizationMethod
(M062)            Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M063)          <ds:SignatureMethod
(M064)            Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
(M065)          <ds:Reference URI="#STR1">
(M066)           <ds:Transforms>
(M067)            <ds:Transform
(M068)              Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-
(M069)     200401-wss-soap-message-security-1.0#STR-Transform">
(M070)            <wsse:TransformationParameters>
(M071)             <ds:CanonicalizationMethod
(M072)               Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M073)            </wsse:TransformationParameters>
(M074)            </ds:Transform>
(M075)           </ds:Transforms>
(M076)           <ds:DigestMethod
(M077)             Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(M078)           <ds:DigestValue>...</ds:DigestValue>
(M079)          </ds:Reference>
(M080)          <ds:Reference URI="#MsgBody">
(M081)           <ds:DigestMethod
(M082)             Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(M083)           <ds:DigestValue>...</ds:DigestValue>
(M084)          </ds:Reference>
(M085)          <ds:Reference URI="#timestamp">
(M086)           <ds:DigestMethod
(M087)             Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(M088)           <ds:DigestValue>...</ds:DigestValue>
(M089)          </ds:Reference>
(M090)         </ds:SignedInfo>
(M091)         <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
(M092)         <ds:KeyInfo>
(M093)           <wsse:SecurityTokenReference wsu:id="STR2">
(M094)             <wsse:Reference wsu:id="..." URI="#attesterCert"/>
(M095)           </wsse:SecurityTokenReference>
(M096)         </ds:KeyInfo>
(M097)        </ds:Signature>
(M098)       </wsse:Security>
(M099)      </S12:Header>
(M0100)     <S12:Body wsu:Id="MsgBody">
(M0101)      <ReportRequest>
(M0102)       <TickerSymbol>SUNW</TickerSymbol>
(M0103)      </ReportRequest>
(M0104)     </S12:Body>
(M0105)    </S12:Envelope>
```

Note: For the instructional purposes of this document, in order to be compliant with WS-SecurityPolicy, this copy of the original message had to be modified from the original. In particular, the modifications that are applied above are the inclusion of a wsu:Id attribute in the Timestamp element start tag (M014) and the addition of a ds:Reference element and URI attribute identifying that Timestamp element in the message signature (M085)-(M089). (The original message in

[WSS10-SAML11-INTEROP scenario #3] is not compliant with WS-SecurityPolicy because the Timestamp that it contains is not covered by the message signature in that document.)

Lines (M002)-(M0105) contain the SOAP:Envelope, i.e. the whole SOAP message.

Lines (M012)-(M099) contain the SOAP:Header, which is the header portion of the SOAP message.

Lines (M0100)-(M0104) contain the SOAP:Body, which is just some dummy content for test purposes.

Lines (M013)-(M098) contain the wsse:Security header, which is the primary focus of this example.

Lines (M014)-(M016) contain the wsu:Timestamp, which is required by the policy (P035).

Lines (M017)-(M044) contain the SAML Assertion, which is the sp:SamlToken required by the policy sp:SignedSupportingTokens (P046)-(P050).

Lines (M021)-(M022) identify the SAML Assertion as being Version 1.1 as required by the policy (P048).

Line (M035) identifies the SAML Assertion as having ConfirmationMethod sender-vouches, which is the implicit requirement of the sp:SamlToken being in the SignedSupportingTokens as described above in the policy section covering (P044)-(P052).

Lines (M045)-(M050) contain a wsse:SecurityTokenReference which contains a wsse:KeyIdentifier, which is used to reference the SAML Assertion, as described in [WSS10-SAML11-PROFILE] and required by the policy (P041).

Lines (M051)-(M058) contain the Initiator's wsse:BinarySecurityToken, which is required by the policy (P007)-(P015), and in particular lines (M053)-(M054) identify the token as an X.509 V3 token as required by the policy (P011). Line (M057) contains a truncated portion of the Base64 representation of the actual certificate, which is included in the message as required by the policy sp:IncludeToken (P009).

Lines (M059)-(M0102) contain the ds:Signature, which is the sp: message signature as required by various parts of the Endpoint Policy (P001)-(P055) and Input Message Policy (P057)-(P068).

Lines (M060)-(M095) contain the ds:SignedInfo element which describes the signing algorithm used (M064) and specific elements that are signed (M065)-(M094) as required by the policies, which are described in detail immediately following.

Lines (M061)-(M062) contain the ds:CanonicalizationMethod Algorithm, which is specified as xml-exc-c14n#, which is the default value required by the policy sp:AlgorithmSuite (P025)-(P029).

Line (M064) identifies the SignatureMethod Algorithm as ds:rsa-sha1, which is the asymmetric key signing algorithm required by the policy sp:AlgorithmSuite (P025)-(P029).

Lines (M065)-(M079) identify the SAML Assertion (M017)-(M044) as an element that is signed, which is referenced through the URI "#STR1" on line (M065), which references the SecurityTokenReference (M045)-(M050), which in turn uses the identifier on line (M048) to reference the actual SAML Assertion by its AssertionID attribute on line (M018). The SAML Assertion is required to be signed because it is identified in the policy within the SignedSupportingTokens element on line (P048).

Lines (M077)-(M078) contain the digest algorithm, which is specified to be sha1, as required by the policy sp:Basic256 assertion (P027) in the sp:AlgorithmSuite.

Lines (M080)-(M084) identify the SOAP:Body (M0100)-(M0104) as an element that is signed, which is referenced through the URI "#MsgBody" on line (M080). The SOAP Body is required to be signed by the input message policy lines (P060)-(P062).

Lines (M085)-(M089) identify the wsu:Timestamp (M014)-(M016) as an element that is signed, which is referenced throught the URI "#timestamp" on line (M085). The wsu:Timestamp is required to be signed by the policy sp:IncludeTimestamp assertion (P035). Note that the wsu:Timestamp occurs in the wsse:Security header prior to the ds:Signature as required by the sp:Strict layout policy (P032).

Finally, lines (M092)-(M096) contain the ds:KeyInfo element that identifies the signing key associated with this ds:Signature element. The actual signing key is referenced through the wsse:SecurityTokenReference (M093)-(M095), with URI "#attesterCert", which identifies the InitiatorToken identified by the policy (P011) and contained in the wsse:BinarySecurityToken element (M051)-(M058), which occurs in the wsse:Security header prior to this ds:Signature element, as required by the sp:Strict layout policy assertion (P032). (Note: this BinarySecurityToken would need to be included in the signature, if the sp:AsymmetricBinding assertion in the endpoint policy contained a sp:ProtectTokens assertion, but it does not, so it does not need to be signed.)

2.3.1.5 (WSS1.0) SAML1.1 Holder of Key, Sign, Optional Encrypt

This example is based on the first WSS SAML Profile InterOp [WSS10-SAML11-INTEROP Scenario #4].

In this example the SAML token provides the key material for message security, hence acts as the Initiator token, and therefore the Requestor and Initiator may be considered to be the same entity. The SAML HK Assertion contains a reference to the public key of the signer of the message (the Initiator). The Initiator knows Recipient's public key, which it may use to encrypt the request, but the Initiator does not share a direct trust relation with the Recipient except indirectly throught the SAML Assertion Issuer. The Recipient has a trust relation with the Authority that issues the SAML HK Assertion The indirect trust relation between the Recipient and the Initiator is established by the fact that the Initiator’s public key has been signed by the Authority in the SAML HK Assertion. On the request the message body is signed using Initiator’s private key referenced in the SAML HK Assertion and it is optionally encrypted using Recipient’s server certificate. On the response, the server signs the message using its private key and encrypts the message using the key provided within SAML HK Assertion.

> HK Note: there is a trust model aspect to the WS-Security holder-of-key examples that implementors may want to be aware of. The [SAML20-CORE] specification defines the Subject of the SAML Assertion such that the “presenter” of the Assertion is the entity that “attests” to the information in the Assertion. “The attesting entity and the actual Subject **may or may not** be the same entity.” In general, these two choices map to the actors in Figure 1 as follows: the “attesting” entity may be regarded as the Initiator. The Subject entity, about whom the information in the Assertion applies, may be regarded as the Requestor. In the case where the Subject and attestor are one and the same, the Initiator and Requestor may be regarded as one and the same. In this case the potential exists to use the Assertion for non-repudiation purposes with respect to messages that the Requestor/Initiator signs. When the Subject and attestor are separate entities, then holder-of-key is more similar to sender-vouches where the Initiator/attestor is sending the message on behalf of the Subject. The mechanisms for determining whether the Subject and attestor may be verified to be the same entity are dependent on the arrangements among the business entities and the structure of the associated application scenarios.

```
(P001)   <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
(P002)       wsu:Id="Wss10SamlHkV11Asymm_endpoint_policy">
(P003)     <wsp:ExactlyOne>
(P004)       <wsp:All>
(P005)         <sp:AsymmetricBinding>
(P006)           <wsp:Policy>
(P007)             <sp:InitiatorToken>
(P008)               <wsp:Policy>
(P009)                 <sp:SamlToken sp:IncludeToken="http://docs.oasis-
         open.org/ws-sx/ws-
         securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P010)                   <wsp:Policy>
(P011)                     <wsp:WssSamlV11Token10/>
(P012)                   </wsp:Policy>
(P013)                 </sp:SamlToken>
(P014)               </wsp:Policy>
(P015)             </sp:InitiatorToken>
(P016)             <sp:RecipientToken>
(P017)               <wsp:Policy>
(P018)                 <sp:X509Token sp:IncludeToken="http://docs.oasis-
         open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
(P019)                   <wsp:Policy>
(P020)                     <sp:WssX509V3Token10/>
(P021)                   </wsp:Policy>
(P022)                 </sp:X509Token>
(P023)               </wsp:Policy>
(P024)             </sp:RecipientToken>
```

```
(P025)             <sp:AlgorithmSuite>
(P026)               <wsp:Policy>
(P027)                 <sp:Basic256/>
(P028)               </wsp:Policy>
(P029)             </sp:AlgorithmSuite>
(P030)             <sp:Layout>
(P031)               <wsp:Policy>
(P032)                 <sp:Strict/>
(P033)               </wsp:Policy>
(P034)             </sp:Layout>
(P035)             <sp:IncludeTimestamp/>
(P036)             <sp:OnlySignEntireHeadersAndBody/>
(P037)           </wsp:Policy>
(P038)         </sp:AsymmetricBinding>
(P039)         <sp:Wss10>
(P040)           <wsp:Policy>
(P041)             <sp:MustSupportRefKeyIdentifier/>
(P042)           </wsp:Policy>
(P043)         </sp:Wss10>
(P044)       </wsp:All>
(P045)     </wsp:ExactlyOne>
(P046)   </wsp:Policy>
(P047)
(P048)   <wsp:Policy wsu:Id="WSS10SamlHok_input_policy">
(P049)     <wsp:ExactlyOne>
(P050)       <wsp:All>
(P051)         <sp:SignedParts>
(P052)           <sp:Body/>
(P053)         </sp:SignedParts>
(P054)         <wsp:ExactlyOne>
(P055)           <wsp:All>
(P056)             <sp:EncryptedParts>
(P057)               <sp:Body/>
(P058)             </sp:EncryptedParts>
(P059)           </wsp:All>
(P060)           <wsp:All/>
(P061)         </wsp:ExactlyOne>
(P062)       </wsp:All>
(P063)     </wsp:ExactlyOne>
(P064)   </wsp:Policy>
(P065)
(P066)   <wsp:Policy wsu:Id="WSS10SamlHok_output_policy">
(P067)     <wsp:ExactlyOne>
(P068)       <wsp:All>
(P069)         <sp:SignedParts>
(P070)           <sp:Body/>
(P071)         </sp:SignedParts>
(P072)         <sp:EncryptedParts>
(P073)           <sp:Body/>
(P074)         </sp:EncryptedParts>
(P075)       </wsp:All>
(P076)     </wsp:ExactlyOne>
(P077)   </wsp:Policy>
```

Lines (P001)-(P046) contain the endpoint policy, and lines (P048)-(P064) and (P066)-(P077) contain the message input and message output policies, respectively.

Lines (P005)-(P038) contain the AsymmetricBinding assertion, which requires separate security tokens for the Initiator and Recipient.

Lines (P007)-(P015) contain the InitiatorToken, which is defined to be a SamlToken (P009)-(P013), which must always be sent to the Recipient (P009). The SamlToken is further specified by the WssSamlV11Token10 assertion (P011) that requires the token to be a SAML 1.1 Assertion and the token must be used in accordance with the WS-Security [WSS10-SAML11-PROFILE]. Furthermore, as described in section 2.3 above, because the SamlToken assertion appears within an InitiatorToken assertion, the SAML Assertion must use the holder-of-key ConfirmationMethod, whereby for the indicated WS-Security profile, the SAML Assertion contains a reference to the Initiator's X.509 signing certificate or equivalent.

Lines (P016)-(P024) contain the RecipientToken assertion. Again, this an X.509 V3 certificate (P020) that must be used as described in [WSS10-SAML11-PROFILE], however the token itself, must never be included in messages in either direction (P018) (i.e not only will the Recipient not send the token, but the Initiator must not send the actual token, even when using it for encryption).

Lines (P025)-(P029) AlgorithmSuite and (P030)-(P034) Layout are the same as described above in section 2.3.1.4.

Line (P035) IncludeTimestamp means a Timestamp element must be included in the WS-Security header block and signed by the message signature for messages in both directions.

Line (P036) OnlySignEntireHeaderAndBody indicates that the message signature must explicitly cover the SOAP:Body element (not children), and if there are any signed elements in the SOAP:Header they must be direct child elements of the SOAP:Header. However, the one exception where the latter condition is not applied is that if the child of the SOAP:Header is a WS-Security header (i.e. a wsse:Security element), then individual direct child only elements of that Security element may also be signed.

Lines (P039)-(P043) contain the WSS10 assertion, which indicates that the Recipient's token must be referenced by a KeyIdentifier element (P041), the usage of which is described in the WS-Security 1.0 core specification [WSS10-SOAPMSG].

Lines (P048)-(P064) contain the message input policy that applies only to messages from the Initiator to the Recipient.

Lines (P051)-(P053) specify that the SOAP:Body element of the input message must be signed by the Initiator's message signature.

Lines (P054)-(P061) specify that the SOAP:Body element of the input message may optionally (signified by the empty policy alternative on line (P060)) be encrypted using the Recipient's encryption token (in this case (P020), it is an X.509 certificate).

> Note: Because the input policy above (P048-P064) has the EncryptedParts assertion (P056-P058) contained in an <ExactlyOne> element (P054-P061), which also contains an empty policy element (P060), either a message with an encrypted Body element or an unencrypted Body element will be accepted.

Lines (P066)-(P077) contain a message output policy that applies only to messages from the Recipient to the Initiator.

Lines (P069)-(P071) specify that the SOAP:Body element of the output message must be signed by the Recipient's message signature.

Lines (P072)-(P074) specify that the SOAP:Body element of the output message must be encrypted by the Initiator's encryption token (in this case (P012), it is also an X.509 certificate).

The following example request is taken from the [WSS10-SAML11-INTEROP] document scenario #4:

```
(M001)   <?xml version="1.0" encoding="utf-8" ?>
(M002)   <S12:Envelope
(M003)     xmlns:S12=http://schemas.xmlsoap.org/soap/envelope/
(M004)     xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
(M005)     xmlns:xsd=http://www.w3.org/2001/XMLSchema
(M006)     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401
(M007)   wss-wssecurity-secext-1.0.xsd"
(M008)     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss
(M009)   wssecurity-utility-1.0.xsd"
(M010)     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
(M011)    <S12:Header>
```

```
(M012)      <wsse:Security S12:mustUnderstand="1">
(M013)       <wsu:Timestamp wsu:Id="timestamp">
(M014)         <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
(M015)       </wsu:Timestamp>
(M016)       <saml:Assertion
(M017)         AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
(M018)         IssueInstant="2003-04-17T00:46:02Z"
(M019)         Issuer=www.opensaml.org
(M020)         MajorVersion="1"
(M021)         MinorVersion="1"
(M022)         xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
(M023)        <saml:Conditions
(M024)          NotBefore="2002-06-19T16:53:33.173Z"
(M025)          NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
(M026)        <saml:AttributeStatement>
(M027)         <saml:Subject>
(M028)          <saml:NameIdentifier
(M029)            NameQualifier=www.example.com
(M030)            Format="">
(M031)           uid=joe,ou=people,ou=saml-demo,o=example.com
(M032)          </saml:NameIdentifier>
(M033)          <saml:SubjectConfirmation>
(M034)           <saml:ConfirmationMethod>
(M035)            urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
(M036)           </saml:ConfirmationMethod>
(M037)           <ds:KeyInfo>
(M038)            <ds:KeyValue>...</ds:KeyValue>
(M039)           </ds:KeyInfo>
(M040)          </saml:SubjectConfirmation>
(M041)         </saml:Subject>
(M042)         <saml:Attribute
(M043)           AttributeName="MemberLevel"
(M044)           AttributeNamespace=
(M045)             "http://www.oasis-open.org/Catalyst2002/attributes">
(M046)          <saml:AttributeValue>gold</saml:AttributeValue>
(M047)         </saml:Attribute>
(M048)         <saml:Attribute
(M049)           AttributeName="E-mail"
(M050)           AttributeNamespace="http://www.oasis-
(M051)           open.org/Catalyst2002/attributes">
(M052)          <saml:AttributeValue>joe@yahoo.com</saml:AttributeValue>
(M053)         </saml:Attribute>
(M054)        </saml:AttributeStatement>
(M055)       <ds:Signature>...</ds:Signature>
(M056)      </saml:Assertion>
(M057)      <ds:Signature>
(M058)       <ds:SignedInfo>
(M059)        <ds:CanonicalizationMethod
(M060)          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M061)        <ds:SignatureMethod
(M062)          Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
(M063)        <ds:Reference URI="#MsgBody">
(M064)         <ds:DigestMethod
(M065)           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(M066)         <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
(M067)        </ds:Reference>
(M068)        <ds:Reference URI="#timestamp">
(M069)         <ds:DigestMethod
(M070)           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(M071)         <ds:DigestValue>...</ds:DigestValue>
(M072)        </ds:Reference>
(M073)       </ds:SignedInfo>
(M074)       <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
(M075)        <ds:KeyInfo>
```

```
(M076)          <wsse:SecurityTokenReference wsu:id="STR1">
(M077)           <wsse:KeyIdentifier wsu:id="..."
(M078)             ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-
(M079)    2004XX-wss-saml-token-profile-1.0#SAMLAssertionID">
(M080)              _a75adf55-01d7-40cc-929f-dbd8372ebdfc
(M081)           </wsse:KeyIdentifier>
(M082)           </wsse:SecurityTokenReference>
(M083)          </ds:KeyInfo>
(M084)         </ds:Signature>
(M085)        </wsse:Security>
(M086)       </S12:Header>
(M087)      <S12:Body wsu:Id="MsgBody">
(M088)       <ReportRequest>
(M089)        <TickerSymbol>SUNW</TickerSymbol>
(M090)       </ReportRequest>
(M091)    </S12:Body>
(M092)    </S12:Envelope>
```

Note: for the instructional purposes of this document, it was necessary to make changes to the original sample request to meet the requirements of WS-SecurityPolicy. The changes to the message above include:

- Line (M013): added wsu:Id="timestamp" attribute to sp:Timestamp element.
- Lines (M068)-(M072): added a ds:Reference element to include the Timestamp in the message signature required by the policy sp:IncludeTimestamp assertion (P035).

Lines (M002)-(M092) contain the SOAP Envelope, i.e. the whole SOAP message.
Lines (M011)-(M086) contain the SOAP Header.
Lines (M087)-(M091) contain the unencrypted SOAP Body, which is allowed to be unencrypted based on line (P060) of the input message policy, which is the alternative choice to encrypting based on lines (P055)-(P059).
Lines (M012)-(M080) contain the wsse:Security header entry, which is the only header entry in this SOAP Header.
Lines (M013)-(M015) contain the wsu:Timestamp which is required by the endpoint policy (P035).
Lines (M016)-(M056) contain the SAML Assertion, which is required in the policy by the SamlToken (P009)-(P013) contained in the InitiatorToken. The SAML Assertion is version 1.1 (M020)-(M021) as required by the sp:WssSamlV11Token10 assertion (P011). The SAML Assertion is of type that uses the holder-of-key saml:ConfirmationMethod [SAML11-CORE] (M034)-(M036) as required by the fact that the SamlToken is contained in the InitiatorToken assertion of the policy (P009)-(P013), as explained in section 2.3 above.
Line (M055) contains the ds:Signature of the Issuer of the SAML Assertion. While the details of this signature are not shown, it is this signature that the Recipient must ultimately trust in order to trust the rest of the message.
Lines (M057)-(M084) contain the message signature in a ds:Signature element.
Lines (M058)-(M073) contain the ds:SignedInfo element, which identifies the elements to be signed.
Lines (M063)-(M067) contains a ds:Reference to the SOAP:Body, which is signed in the same manner as example section 2.3.1.4 above.
Lines (M068)-(M072) contains a ds:Reference to the URI "timestamp", which again is signed in the same manner as the wsu:Timestamp in section 2.3.1.4 above.
Lines (M075)-(M083) contain the ds:KeyInfo element, which identifies the key that should be used to verify this ds:Signature element. Lines (M076)-(M082) contain a wsse:SecurityTokenReference, which contains a wsse:KeyIdentifier that identifies the signing key as being contained in a token of wsse:ValueType "…wss-saml-token-profile-1.0#SAMLAssertionID", which means a SAML V1.1 Assertion [WSS10-SAML11-PROFILE]. Line (M080) contains the saml:AssertionID of the SAML Assertion that contains the signing key. This SAML Assertion is on lines (M016)-(M056) with the correct saml:AssertionID on line (M017). Note that the fact that the referenced token (the SAML Assertion) occurs in the wsse:Security header before this ds:KeyInfo element that uses it in compliance with the sp:Strict layout policy (P032) and the wsse:KeyIdentifier is an acceptable token reference mechanism as specified in the policy sp:Wss10 assertion containing a sp:MustSupportRefKeyIdentifier assertion (P041).

2.3.2 WSS 1.1 SAML Token Scenarios

This section contains SamlToken examples that use WS-Security 1.1 SOAP Message Security [WSS11] and the WS-Security 1.1 SAML Profile [WSS11-SAML1120-PROFILE].

2.3.2.1 (WSS1.1) SAML 2.0 Bearer

This example is based on the Liberty Alliance Identity Web Services Framework (ID-WSF 2.0) Security Mechanism for the SAML 2.0 Profile for WSS 1.1 [WSS11-LIBERTY-SAML20-PROFILE], which itself is based on the [WSS11-SAML1120-PROFILE].

In this example, an AsymmetricBinding is used for message protection provided by the Initiator, however, Recipient trust for the content is based only on the SAML bearer token provided by the Requestor.

```
(P001)    <wsp:Policy>
(P002)      <sp:AsymmetricBinding>
(P003)        <wsp:Policy>
(P004)          <sp:InitiatorToken>
(P005)            <wsp:Policy>
(P006)              <sp:X509Token sp:IncludeToken="http://docs.oasis-
          open.org/ws-sx/ws-
          securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P007)                <wsp:Policy>
(P008)                  <sp:WssX509V3Token10/>
(P009)                </wsp:Policy>
(P010)              </sp:X509Token>
(P011)            </wsp:Policy>
(P012)          </sp:InitiatorToken>
(P013)          <sp:RecipientToken>
(P014)            <wsp:Policy>
(P015)              <sp:X509Token sp:IncludeToken="http://docs.oasis-
          open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
(P016)                <wsp:Policy>
(P017)                  <sp:WssX509V3Token10/>
(P018)                </wsp:Policy>
(P019)              </sp:X509Token>
(P020)            </wsp:Policy>
(P021)          </sp:RecipientToken>
(P022)          <sp:AlgorithmSuite>
(P023)            <wsp:Policy>
(P024)              <sp:Basic256/>
(P025)            </wsp:Policy>
(P026)          </sp:AlgorithmSuite>
(P027)          <sp:Layout>
(P028)            <wsp:Policy>
(P029)              <sp:Strict/>
(P030)            </wsp:Policy>
(P031)          </sp:Layout>
(P032)          <sp:IncludeTimestamp/>
(P033)          <sp:OnlySignEntireHeadersAndBody/>
(P034)        </wsp:Policy>
(P035)      </sp:AsymmetricBinding>
(P036)      <sp:Wss11>
(P037)        <wsp:Policy>
(P038)          <sp:MustSupportRefKeyIdentifier/>
(P039)          <sp:MustSupportRefIssuerSerial/>
(P040)          <sp:MustSupportRefThumbprint/>
(P041)          <sp:MustSupportRefEncryptedKey/>
(P042)        </wsp:Policy>
(P043)      </sp:Wss11>
(P044)      <sp:SupportingTokens>
(P045)        <wsp:Policy>
```

```
(P046)         <sp:SamlToken sp:IncludeToken=”http://docs.oasis-open.org/ws-
         sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient”>
(P047)           <wsp:Policy>
(P048)             <sp:WssSamlV20Token11/>
(P049)           </wsp:Policy>
(P050)         </sp:SamlToken>
(P051)       </wsp:Policy>
(P052)     </sp:SupportingTokens>
(P053)   </wsp:Policy>
```

Lines (P001)-(P045) contain the endpoint policy, which contains 3 assertions: an sp:AsymmetricBinding assertion, an sp:Wss11 assertion, and an sp:SupportingTokens assertion.

Lines (P002)-(P035) contain the sp:AsymmetricBinding assertion, which is identical to the same assertion that is described in section 2.1.3. Please refer to section 2.1.3 for details.

Lines (P036)-(P043) contain an sp:Wss11 assertion, which contains a set of assertions that specify the token referencing techniques that are required to be supported (P038)-(P041).

Lines (P044)-(P052) contain an sp:SupportingTokens assertion, which contains an sp:SamlToken (P046)-(P050), which specifies that it must always be sent to the Recipient. The sp:SamlToken contains an sp:WssSamlV20Token11 assertion (P048), which means that the SamlToken provided must be a SAML Version 2.0 Assertion that is submitted in compliance with the [WSS11-SAML1120-PROFILE].

The fact that the sp:SamlToken is contained in an sp:SupportingTokens element indicates to the Initiator that the SAML Assertion must use the saml2:SubjectConfirmation@Method “bearer” as described above in introductory section 2.3.

The following is an example request compliant with the above policy:

```
(M001)   <?xml version="1.0" encoding="UTF-8"?>
(M002)   <s:Envelope xmlns:s=".../soap/envelope/" xmlns:sec="..."
(M003)       xmlns:wsse="..." xmlns:wsu="..." xmlns:wsa="...addressing"
(M004)       xmlns:sb="...liberty:sb" xmlns:pp="...liberty.id-sis-pp"
(M005)       xmlns:ds="...xmldsig#" xmlns:xenc="...xmlenc#">
(M006)     <s:Header>
(M007)       <!-- see Liberty SOAP Binding Spec for reqd, optional hdrs  -->
(M008)       <wsa:MessageID wsu:Id="mid">...</wsa:MessageID>
(M009)       <wsa:To wsu:Id="to">...</wsa:To>
(M010)       <wsa:Action wsu:Id="action">...</wsa:Action>
(M011)       <wsse:Security mustUnderstand="1">
(M012)         <wsu:Timestamp wsu:Id="ts">
(M013)           <wsu:Created>2005-06-17T04:49:17Z</wsu:Created >
(M014)         </wsu:Timestamp>
(M015)         <!-- this is the bearer token -->
(M016)         <saml2:Assertion xmlns:saml2="...SAML:2.0:assertion"
(M017)             Version="2.0" ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
(M018)             IssueInstant="2005-04-01T16:58:33.173Z">
(M019)           <saml2:Issuer>http://authority.example.com/</saml2:Issuer>
(M020)           <!-- signature by the issuer over the assertion -->
(M021)           <ds:Signature>...</ds:Signature>
(M022)           <saml2:Subject>
(M023)             <saml2:EncryptedID>
(M024)               <xenc:EncryptedData
(M025)                 >U2XTCNvRX7Bl1NK182nmY00TEk==</xenc:EncryptedData>
(M026)               <xenc:EncryptedKey>...</xenc:EncryptedKey>
(M027)             </saml2:EncryptedID>
(M028)             <saml2:SubjectConfirmation
(M029)                 Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
(M030)           </saml2:Subject>
(M031)           <!-- audience restriction on assertion can limit scope of
(M032)           which entity should consume the info in the assertion. -->
(M033)           <saml2:Conditions NotBefore="2005-04-01T16:57:20Z"
(M034)               NotOnOrAfter="2005-04-01T21:42:4 3Z">
(M035)             <saml2:AudienceRestrictionCondition>
```

```
(M036)               <saml2:Audience>http://wsp.example.com</saml2:Audience>
(M037)             </saml2:AudienceRestrictionCondition>
(M038)           </saml2:Conditions>
(M039)           <!-- The AuthnStatement carries info that describes authN
(M040)           event of the Subject to an Authentication Authority -->
(M041)           <saml2:AuthnStatement AuthnInstant="2005-04-01T16:57:30.000Z"
(M042)               SessionIndex="6345789">
(M043)             <saml2:AuthnContext>
(M044)               <saml2:AuthnContextClassRef
(M045)               >...:SAML:2.0:ac:classes:PasswordProtectedTransport
(M046)               </saml2:AuthnContextClassRef>
(M047)             </saml2:AuthnContext>
(M048)           </saml2:AuthnStatement>
(M049)           <!-- This AttributeStatement carries an EncryptedAttribute.
(M050)           Once this element decrypted with supplied key an <Attribute>
(M051)           element bearing endpoint reference can be found, specifying
(M052)           resources which invoker may access. Details on element can be
(M053)           found in discovery service specification. -->
(M054)           <saml2:AttributeStatement>
(M055)             <saml2:EncryptedAttribute>
(M056)               <xenc:EncryptedData
(M057)                   Type="http://www.w3.org/2001/04/xmlenc#Element" >
(M058)                 mQEMAzRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzqXdgcX8
(M059)                 ... hg6nZ5c0I6L6Gn9A=HCQY
(M060)               </xenc:EncryptedData>
(M061)               <xenc:EncryptedKey> ... </xenc:EncryptedKey>
(M062)             </saml2:EncryptedAttribute>
(M063)           </saml2:AttributeStatement>
(M064)         </saml2:Assertion>
(M065)         <!-- This SecurityTokenReference is used to reference the SAML
(M066)         Assertion from a ds:Reference -->
(M067)         <wsse:SecurityTokenReference xmlns:wsse="..." xmlns:wsu="..."
(M068)             xmlns:wsse11="..." wsu:Id="str1" wsse11:TokenType=
(M069)               ".../wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
(M070)           <wsse:KeyIdentifier ValueType=
(M071)               ".../wss/oasis-wss-sam l-token-profile-1.1#SAMLID"
(M072)           >sxJu9g/vvLG9sAN9bKp/8q0NKU=</wsse:KeyIdentifier>
(M073)         </wsse:SecurityTokenReference>
(M074)         <ds:Signature>
(M075)           <ds:SignedInfo>
(M076)           <!-- in general include a ds:Reference for each wsa:header
(M077)           added according to SOAP binding, plus timestamp, plus ref to
(M078)           assertion to avoid token substitution attacks, plus Body -->
(M079)             <ds:Reference URI="#to">...</ds:Reference>
(M080)             <ds:Reference URI="#action">...</ds:Reference>
(M081)             <ds:Reference URI="#mid">...</ds:Reference>
(M082)             <ds:Reference URI="#ts">...</ds:Reference>
(M083)             <ds:Reference URI="#Str1">
(M084)               <ds:Transform Algorithm="...#STR-Transform">
(M085)                 <wsse:TransformationParameters>
(M086)                   <ds:CanonicalizationMethod Algorithm=
(M087)                     "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
(M088)                 </wsse:TransformationParameters>
(M089)               </ds:Transform>
(M090)             </ds:Reference>
(M091)             <ds:Reference URI="#MsgBody">...</ds:Reference>
(M092)           </ds:SignedInfo>
(M093)             ...
(M094)         </ds:Signature>
(M095)       </wsse:Security>
(M096)     </s:Header>
(M097)     <s:Body wsu:Id="MsgBody">
(M098)       <pp:Modify>
(M099)         <!-- this is an ID-SIS-PP Modify message -->
```

```
(M0100)       </pp:Modify>
(M0101)     </s:Body>
(M0102)   </s:Envelope>
```

The sample message above was taken and cosmetically edited from the [WSS11-LIBERTY-SAML20-PROFILE].

Lines (M002)-(M0102) contain the SOAP:Envelope, which contains the SOAP:Header (M006)-(M096) and the SOAP:Body (M097)-(M0101).

The SOAP Header contains some WS-Addressing (wsa:) parameters (M008)-(M010) (not discussed here) and a wsse:Security header.

The wsse:Security header (M011)-(M095) contains a wsu:Timestamp (M012)-(M014), a saml2:Assertion (M016)-(M064), a wsse:SecurityTokenReference (M067)-(M073), and a ds:Signature (M074)-(M094).

The wsu:Timestamp (M012)-(M014) is required by the policy sp:IncludeTimestamp assertion (P032).

The saml2:Assertion (M016)-(M064) is required by the policy sp:WssSamlV20Token11 (P048). The saml2:Assertion is Version 2.0 (M017) and it is signed by the IssuingAuthority (M021). This is the ds:Signature (M021) of the IssuingAuthority, identified in the saml2:Issuer element (M019), that the Recipient trusts with respect to recognizing the Requestor and determining whether the request should be granted. In addition, this saml2:Assertion uses the “bearer” saml2:SubjectConfirmation@Method, as required by the policy sp:SamlToken in the sp:SupportingTokens element described above (P044)-(P052).

> Note: the saml2:Assertion contains a saml2:EncryptedID (M023)-(M027), which contains the identity of the Requestor and a saml2:EncryptedAttribute (M062), which contains information about the Requestor. It is presumed that prior to issuing this request, that the Requestor contacted the IssuingAuthority (directly or indirectly) and was granted permission to access the web service that is now the target of this request. As such, the IssuingAuthority has knowledge of this web service and presumably the public certificate associated with that service and has used the public key contained in that certificate to encrypt the aforementioned portions of the saml2:Assertion, which can only be decrypted by the RelyingParty who presumably has entrusted the private key of the service with the Recipient, in order that the Recipient may decrypt the necessary data.

However, before the Recipient can evaluate these aspects of the request, the requirements of the policy sp:AsymmetricBinding (P002)-(P035) must be met in terms of proper “presentation” of the request as described below.

Lines (M074)-(M094) contain the message signature ds:Signature element, which contains a ds:SignedInfo that contains ds:Reference elements that cover the wsa: headers (M079)-(M081), the wsu:Timestamp (M082) (required by the policy sp:IncludeTimestamp assertion (P032), the saml2:Assertion (M083)-(M090), and the SOAP:Body (M091).

The ds:Reference (M083)-(M090) covering the saml2:Assertion warrants further examination.

> The first point to note is that to reference the saml2:Assertion the ds:Reference uses an STR-Transform (M084) to reference a wsse:SecurityTokenReference (M067)-(M073), which is in compliance with policy sp:Wss11 sp:MustSupportRefKeyIdentifier assertion (P038).
> The second point to note about this ds:Reference is that is covering the saml2:Assertion even though the policy references the sp:SamlToken in an sp:SupportingTokens element and not an sp:SignedSupportingTokens element. The reason this is the case is that it is the fact that an sp:SupportingTokens (P044)-(P052) element is used that tells the Initiator that a SAML bearer Assertion is required. However, this only means that the SamlToken is not “required” to be signed. It is the Initiator’s choice whether to sign it or not, and it is generally good practice to do so in order to prevent a token substitution attack.

2.3.2.2 (WSS1.1) SAML2.0 Sender Vouches over SSL

This scenario is based on second WSS SAML Profile InterOp [WSS11-SAML1120-INTEROP Scenario #2].

Similar to 2.3.1.2 except SAML token is of version 2.0.

```
(P001)   <wsp:Policy>
(P002)     <sp:TransportBinding>
(P003)       <wsp:Policy>
(P004)         <sp:TransportToken>
(P005)           <wsp:Policy>
(P006)             <sp:HttpsToken>
(P007)               <wsp:Policy>
(P008)                 <sp:RequireClientCertificate>
(P009)               </wsp:Policy>
(P010)             </sp:HttpsToken>
(P011)           </wsp:Policy>
(P012)         </sp:TransportToken>
(P013)         <sp:AlgorithmSuite>
(P014)           <wsp:Policy>
(P015)             <sp:Basic256 />
(P016)           </wsp:Policy>
(P017)         </sp:AlgorithmSuite>
(P018)         <sp:Layout>
(P019)           <wsp:Policy>
(P020)             <sp:Strict />
(P021)           </wsp:Policy>
(P022)         </sp:Layout>
(P023)         <sp:IncludeTimestamp />
(P024)       </wsp:Policy>
(P025)     </sp:TransportBinding>
(P026)     <sp:SignedSupportingTokens>
(P027)       <wsp:Policy>
(P028)         <sp:SamlToken sp:IncludeToken=”http://docs.oasis-open.org/ws-
         sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient”>
(P029)           <wsp:Policy>
(P030)             <sp:WssSamlV20Token11/>
(P031)           </wsp:Policy>
(P032)         </sp:SamlToken>
(P033)       </wsp:Policy>
(P034)     </sp:SignedSupportingTokens>
(P035)   </wsp:Policy>
```

Lines (P001)-(P035) contain the policy that requires all the contained assertions to be complied with by the Initiator. In this case there are 2 assertions: sp:TransportBinding (P002) and sp:SignedSupportingTokens (P026).

Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected by a secure transport protocol such as SSL or TLS.

Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the transport token using the Basic256 algorithm suite (P015).

In addition, because this is SAML sender-vouches, a client certificate is required (P008) as the basis of trust for the SAML Assertion and for the content of the message [WSS10-SAML11-INTEROP section 4.3.1].

The requirements for the sp:Layout assertion (P018)-(P022) should be automatically met (or may be considered moot) since there are no cryptographic tokens required to be present in the WS-Security header. (However, if a signature element was included to cover the wsse:Timestamp, then the layout would need to be considered.)

A timestamp (P023) is required to be included in an acceptable message.

Here is an example request:

```
(M001)   <?xml version="1.0" encoding="utf-8" ?>
(M002)   <S11:Envelope xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
(M003)       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
(M004)       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
(M005)       xmlns:wsu=".../oasis-200401-wsswssecurity-utility-1.0.xsd"
(M006)       xmlns:wsse=".../oasis-200401-wss-wssecurity-secext-1.0.xsd"
(M007)       xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
(M008)     <S11:Header>
(M009)       <wsse:Security S11:mustUnderstand="1">
(M010)         <wsu:Timestamp>
(M011)           <wsu:Created>2005-03-18T19:53:13Z</wsu:Created>
(M012)         </wsu:Timestamp>
(M013)         <saml2:Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
(M014)             IssueInstant="2005-04-17T00:46:02Z" Version="2.0">
(M015)           <saml2:Issuer>www.opensaml.org</saml2:Issuer>
(M016)           <saml2:Conditions NotBefore="2005-06-19T16:53:33.173Z"
(M017)             NotOnOrAfter="2006-06-19T17:08:33.173Z" />
(M018)           <saml2:Subject>
(M019)             <saml2:NameID Format=
(M020)                 "urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
(M021)            >uid=joe,ou=people,ou=saml demo,o=example.com</saml2:NameID>
(M022)             <saml2:SubjectConfirmation Method=
(M023)                 "urn:oasis:names:tc:SAML:2.0:cm:sender-vouches" />
(M024)           </saml2:Subject>
(M025)           <saml2:AttributeStatement>
(M026)             <saml2:Attribute>...</saml2:Attribute>
(M027)             <saml2:Attribute>...</saml2:Attribute>
(M028)           </saml2:AttributeStatement>
(M029)         </saml2:Assertion>
(M030)       </wsse:Security>
(M031)     </S11:Header>
(M032)     <S11:Body>
(M033)       <Ping xmlns="http://xmlsoap.org/Ping">
(M034)         <text>EchoString</text>
(M035)       </Ping>
(M036)     </S11:Body>
(M037)   </S11:Envelope>
```

Lines (M002)-(M037) contain the SOAP:Envelope, which contains the SOAP:Header (M003)-(M031) and the SOAP:Body (M032)-(M036).

Lines (M009)-(M030) contain the wsse:Security header, which, in conjunction with the client certificate (P008), is the primary basis for trust in this example.

Lines (M010)-(M012) contain the wsu:Timestamp, which meets the sp:IncludeTimestamp assertion policy requirement (P023).

Lines (M013)-(M029) contain the saml2:Assertion, which is required to be Version 2.0 (M014) and to be used within the [WSS11-SAML1120-PROFILE], because of the policy sp:SamlToken assertion (P028)-(P032), which contains the sp:WssSamlV20Token11 (P030).

Lines (M022)-(M023) indicate that the SAML Assertion uses the saml2:Method sender-vouches for the purposes of saml2:SubjectConfirmation, which is required by the fact that the sp:SamlToken appears in an sp:SignedSupportingTokens assertion (P026)-(P034) in conjunction with the client certificate required by the sp:RequireClientCertificate asertion (P008) within the sp:TransportBinding as described in section 2.3 above. In addition, the Recipient should be able to correlate the saml2:Issuer (M015) as being properly associated with the client certficate received from the HTTPS (SSL) connection.

2.3.2.3 (WSS1.1) SAML2.0 HoK over SSL

This scenario is based on second WSS SAML Profile InterOp [WSS11-SAML1120-INTEROP Scenario #5].

Similar to 2.3.1.3 except SAML token is of version 2.0.

Initiator adds a SAML Assertion (hk) to the SOAP Security Header. In this policy the sp:TransportBinding requires a Client Certificate AND the sp:SamlToken is in an sp:SignedEndorsingSupportingTokens element. A SAML holder-of-key Assertion meets these requirements because it is “virtually signed” by the message signature as a result of the SSL client certificate authentication procedure as described in section 2.3. Furthermore, the SAML hk Assertion in this case is a “virtually endorsing”, because the key identified in the holder-of-key saml2:SubjectConfirmationData is also the client certificate, which is virtually endorsing its own signature, under the authority of the IssuingAuthority who has signed the SAML hk Assertion.

As a result, the Initiator may be considered to be authorized by the saml2:Issuer of the hk SAML Assertion to bind message content to the Subject of the Assertion. If the Client Certificate matches the certificate identified in the hk Assertion, the Initiator may be regarded as executing SAML hk responsibility of binding the Requestor, who would be the Subject of the hk Assertion, to the content of the message.

> (Note: the same considerations described in section 2.3.1.4 with respect to whether the Subject of the Assertion and the Subject of the Client Certificate are the same entity determining whether the Client Certificate is attesting for the Assertion Subject or whether the Client Certificate is authenticating as the Assertion Subject apply here.)

In this scenario, the IssuingAuthority is the saml2:Issuer(signer) of the hk SAML Assertion. The Requestor is the Subject of the Assertion and the Initiator is authorized by the IssuingAuthority to bind the Assertion to the message using the ClientCertificate identified in the SAML Assertion, which may also be considered to be virtually signing the wsu:Timestamp of the message. Optionally, a separate Signature may be used to sign the wsu:Timestamp, which the Recipient would also be required to verify was signed by the client certificate in this example.

```
<wsp:Policy>
  <sp:TransportBinding>
    <wsp:Policy>
      <sp:TransportToken>
        <wsp:Policy>
          <sp:HttpsToken>
            <wsp:Policy>
              <sp:RequireClientCertificate>
            </wsp:Policy>
          </sp:HttpsToken>
        </wsp:Policy>
      </sp:TransportToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          <sp:Basic256 />
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
    </wsp:Policy>
  </sp:TransportBinding>
  <sp:SignedEndorsingSupportingTokens>
    <wsp:Policy>
      <sp:SamlToken sp:IncludeToken=”http://docs.oasis-open.org/ws-
sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient”>
        <wsp:Policy>
```

```
(P030)              <sp:WssSamlV20Token11/>
(P031)            </wsp:Policy>
(P032)          </sp:SamlToken>
(P033)        </wsp:Policy>
(P034)      </sp:SignedEndorsingSupportingTokens>
(P035)    </wsp:Policy>
```

Lines (P001)-(P035) contain the policy that requires all the contained assertions to be complied with by the Initiator. In this case there are 2 assertions: sp:TransportBinding (P002) and sp:EndorsingSupportingTokens (P026).

Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected by a secure transport protocol such as SSL or TLS.

Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the transport token using the Basic256 algorithm suite (P015).

In addition, because this is SAML holder-of-key, a client certificate is required (P008) as the basis of trust for the SAML Assertion and for the content of the message [WSS10-SAML11-INTEROP section 4.3.1]. In the holder-of-key case, there will be an additional certificate required in the trust chain, which is the certificate used to sign the SAML hk Assertion, which will be contained or referenced in the Assertion.

The layout requirement in this case (P018)-(P022) is automatically met (or may be considered moot) since there are no cryptographic tokens required to be present in the WS-Security header. (However, if a signature element was included to cover the wsse:Timestamp, then the layout would need to be considered.)

A timestamp (P023) is required to be included in an acceptable message.

Lines (P026)-(P034) contain an sp:SignedEndorsingSupportingTokens element, which means that the contained supporting token (the SAML hk Assertion) references a key that will be “signing” (endorsing) the message signature. The token itself may also be considered to be “signed”, because it is contained in the message sent over the SSL link. In the case of sp:TransportBinding, there may be no actual “message signature”, however, when a client certificate is used, the service can be assured that the connection was set up by the client and that the SSL link guarantees the integrity of the data that is sent on the link by the client, while it is on the link and when it is received from the link by using the key referenced by the token. However, it does not guarantee the integrity after the data is received (i.e. after it is received there is no way to tell whether any changes have been made to it since it has been received). In any event, within this context a Signed Endorsing Supporting Token can be used to tell the Recipient that the Issuer of the token is making claims related to the holder of the private key that is referenced by the token, which in this case would be the private key associated with the client certificate used to set up the SSL link, as described further below.

Lines (P028)-(P032) contain an sp:SamlToken, which must always be sent to the Recipient.

Line (P030) specifies that the token is of type WssSamlV20Token11, which means that the Endorsing Supporting Token must be a SAML Version 2.0 token and it must be sent using WS-Security 1.1 using the [WSS11-SAML1120-PROFILE]. Note that because the SamlToken is contained in an sp:EndorsingSupportingTokens element, that it implicitly must be a holder-of-key token as described in section 2.3 above.

Here is an example request taken from the WSS SAML Profile InterOp [WSS11-SAML1120-INTEROP Scenario #5] containing only minor cosmetic modifications and corrections.

```
(M001)    <?xml version="1.0" encoding="utf-8" ?>
(M002)    <S11:Envelope
(M003)        xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
(M004)        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
(M005)        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
(M006)        xmlns:wsu=".../oasis-200401-wss-wssecurity-utility-1.0.xsd"
(M007)        xmlns:wsse=".../oasis-200401-wss-wssecurity-secext-1.0.xsd"
(M008)        xmlns:wsse11=".../oasis-wss-wssecurity-secext-1.1.xsd"
```

```
(M009)       xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
(M010)       xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
(M011)     <S11:Header>
(M012)       <wsse:Security S11:mustUnderstand="1">
(M013)         <wsu:Timestamp>
(M014)           <wsu:Created>2005-03-18T19:53:13Z</wsu:Created>
(M015)         </wsu:Timestamp>
(M016)         <saml2:Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
(M017)             IssueInstant="2005-04-17T00:46:02Z" Version="2.0">
(M018)           <saml2:Issuer>www.opensaml.org</saml2:Issuer>
(M019)       <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
(M020)         <ds:SignedInfo>
(M021)           <ds:CanonicalizationMethod
(M022)               Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M023)           <ds:SignatureMethod
(M024)               Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
(M025)           <ds:Reference URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
(M026)             <ds:Transforms>
(M027)              <ds:Transform Algorithm=
(M028)               "http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
(M029)              <ds:Transform Algorithm=
(M030)                "http://www.w3.org/2001/10/xml-exc-c14n#">
(M031)                 <InclusiveNamespaces
(M032)                   PrefixList="#default saml ds xs xsi"
(M033)                   xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M034)              </ds:Transform>
(M035)             </ds:Transforms>
(M036)             <ds:DigestMethod Algorithm=
(M037)                 "http://www.w3.org/2000/09/xmldsig#sha1"/>
(M038)             <ds:DigestValue
(M039)               >Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
(M040)           </ds:Reference>
(M041)         </ds:SignedInfo>
(M042)         <ds:SignatureValue>
(M043)   hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n
(M044)   7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtp3TD
(M045)   MwuL/cBUj2OtBZOQMFn7jQ9YB7klIz3RqVL+wNmeWI4=
(M046)         </ds:SignatureValue>
(M047)         <ds:KeyInfo>
(M048)           <ds:X509Data>
(M049)             <ds:X509Certificate>
(M050)   MIICyjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakxCzAJBgNVBAYTAlVT
(M051)     ...
(M052)   8I3bsbmRAUg4UP9hH6ABVq4KQKMknxu1xQxLhpR1ylGPdiowMNTrEG8cCx3w/w==
(M053)            </ds:X509Certificate>
(M054)          </ds:X509Data>
(M055)        </ds:KeyInfo>
(M056)      </ds:Signature>
(M057)           <saml2:Conditions NotBefore="2005-06-19T16:53:33.173Z"
(M058)               NotOnOrAfter="2006-06-19T17:08:33.173Z" />
(M059)           <saml2:Subject>
(M060)             <saml2:NameID
(M061)          Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
(M062)          >uid=joe,ou=people,ou=saml-demo,o=example.com</saml2:NameID>
(M063)             <saml2:SubjectConfirmation
(M064)                 Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key"/>
(M065)             <saml2:SubjectConfirmationData
(M066)                 xsi:type="saml2:KeyInfoConfirmationDataType">
(M067)               <ds:KeyInfo>
(M068)                 <ds:X509Data>
(M069)                   <ds:X509IssuerName>
(M070)                     C=ZA, ST=Western Cape, L=Cape Town,
(M071)                     O=Thawte Consulting cc,
(M072)                     OU=Certification Services Division,
(M073)                     CN=Thawte Server CA/Email=server-certs@thawte.com
(M074)                   </ds:X509IssuerName>
(M075)                   <X509SerialNumber>12345678</X509SerialNumber>
(M076)                 </ds:X509Data>
(M077)               </ds:KeyInfo>
(M078)             </saml2:SubjectConfirmationData>
(M079)           </saml2:Subject>
(M080)            <saml2:AttributeStatement>
(M081)              <saml2:Attribute Name="MemberLevel">
(M082)                <saml2:AttributeValue>gold</saml2:AttributeValue>
(M083)              </saml2:Attribute>
(M084)              <saml2:Attribute Name="E-mail">
(M085)                <saml2:AttributeValue
(M086)                  >joe@yahoo.com</saml2:AttributeValue>
(M087)              </saml2:Attribute>
(M088)           </saml2:AttributeStatement>
(M089)         </saml2:Assertion>
(M090)       </wsse:Security>
(M091)     </S11:Header>
(M092)     <S11:Body wsu:Id="MsgBody">
(M093)       <ReportRequest>
(M094)         <TickerSymbol>SUNW</TickerSymbol>
(M095)       </ReportRequest>
(M096)     </S11:Body>
(M097)   </S11:Envelope>
```

Lines (M002)-(M097) contain the SOAP:Envelope, which contains the SOAP:Header (M011)-(M091) and the SOAP:Body (M092)-(M096).

Lines (M012)-(M090) contain the wsse:Security header, which contains a wsu:Timestamp (M013)-(M015) and a saml2:Assertion (M016)-(M089).

The wsu:Timestamp (M013)-(M015) may be considered to be virtually signed by the client certificate as explained above and in section 2.3.

The saml2:Assertion was issued by an IssuingAuthority identified in the saml2:Issuer element (M018).

The saml2:Issuer has used the private key of its enterprise certificate to produce the ds:Signature element (M019)-(M056) that is an enveloped-signature (M028) that covers its parent XML element, the saml2:Assertion (M016)-(M089). The ds:KeyInfo (M047)-(M055) within this ds:Signature contains an actual copy of the Issuer's enterprise certificate, that the Recipient can verify for authenticity to establish that this message is in conformance with any agreement the RelyingParty may have with the IssuingAuthority. The details of this verification are outside the scope of this document, however they involve the structures and systems the RelyingParty has in place recognize and verify certificates and signatures it receives that are associated with business arrangements with the holders of the private keys of those certificates.

The saml2:Subject of the saml2:Assertion is contained in lines (M059)-(M079). Within the saml2:Subject is the saml2:NameID (M060)-(M062), which contains the official name of the Subject of this Assertion and this entity may be considered to the Requestor associated with this request.

The saml2:SubjectConfirmation (M063)-(M064) has Method "holder-of-key" which implies that there is a saml2:SubjectConfirmationData element (M065)-(M078) which will contain information that identifies a signing key that the Initiator will use to bind this saml2:Assertion to a message that is associated with the Requestor. In this context, the Initiator is acting as "attesting entity" with respect to the Subject as defined in [SAML20], which means that the Initiator is authorized by the IssuingAuthority to present saml2:Assertions pertaining to saml2:Subjects to Recipients/RelyingParties. In this example there is a ds:KeyInfo (M067)-(M077) that identifies a specific certificate (ds:X509IssuerName (M069)-(M074) and ds:SerialNumber (M075)) that the Initiator must prove possession of the associated private key to verify this message. In this policy that private key is the one associated with the client certificate that the Initiator is required to use (P008).

The saml2:AttributeStatement (M080)-(M088) contains some information about the Subject that is officially being provided in this saml2:Assertion by the IssuingAuthority that may have some significance to the Relying Party in terms of determining whether access is granted for this request.

2.3.2.4 (WSS1.1) SAML1.1/2.0 Sender Vouches with X.509 Certificate, Sign, Encrypt

Here the message and SAML Assertion are signed using a key derived from the ephemeral key K. The ephemeral key is encrypted using the Recipient's public key for the request input message and the same shared ephemeral key, K, is encrypted using the Initiators public key for the response output message.

Alternatively, derived keys can be used for each of signing and encryption operations.

In this scenario the Authority is the Initiator who signs the message with the generated key. In order to establish trust in the generated key, the Initiator must sign the message signature with a second signature using an X509 certificate, which is indicated as the EndorsingSupportingToken. This X509 certificate establishes the Initiator as the SAML Authority.

(Note: there are instructive similarities and differences between this example and example 2.3.1.4, where the difference is that: here the binding is symmetric and the WSS1.1 is used, whereas in 2.3.1.4 the binding is asymmetric and WSS1.0 is used. One particular item is that in 2.3.1.4 an EndorsingSupportingToken is not needed because in 2.3.1.4 the asymmetric binding uses X.509 certificates, which may be inherently trusted as opposed to the ephemeral key, K, used here.)

```
(P001)   <wsp:Policy wsu:Id="WSS11SamlWithCertificates_policy">
(P002)     <wsp:ExactlyOne>
(P003)       <wsp:All>
(P004)         <sp:SymmetricBinding>
(P005)           <wsp:Policy>
(P006)             <sp:ProtectionToken>
(P007)               <wsp:Policy>
(P008)                 <sp:X509Token sp:IncludeToken=
         "http://docs.oasis-open.org/ws-sx/ws-
         securitypolicy/200702/IncludeToken/Never”>
(P009)                   <wsp:Policy>
(P010)                     <sp:RequireThumbprintReference/>
(P011)                     <sp:RequireDerivedKeys wsp:Optional="true"/>
(P012)                     <sp:WssX509V3Token10/>
(P013)                   </wsp:Policy>
(P014)                 </sp:X509Token>
(P015)               </wsp:Policy>
(P016)             </sp:ProtectionToken>
(P017)             <sp:AlgorithmSuite>
(P018)               <wsp:Policy>
(P019)                 <sp:Basic256/>
(P020)               </wsp:Policy>
(P021)             </sp:AlgorithmSuite>
(P022)             <sp:Layout>
(P023)               <wsp:Policy>
(P024)                 <sp:Strict/>
(P025)               </wsp:Policy>
(P026)             </sp:Layout>
(P027)             <sp:IncludeTimestamp/>
(P028)             <sp:OnlySignEntireHeadersAndBody/>
(P029)           </wsp:Policy>
(P030)         </sp:SymmetricBinding>
(P031)         <sp:SignedSupportingTokens>
(P032)           <wsp:Policy>
(P033)             <sp:SamlToken sp:IncludeToken=
         "http://docs.oasis-open.org/ws-sx/ws-
         securitypolicy/200702/IncludeToken/AlwaysToRecipient">
(P034)               <wsp:Policy>
(P035)                 <sp:WssSamlV11Token11/>
(P036)               </wsp:Policy>
(P037)             </sp:SamlToken>
(P038)           </wsp:Policy>
(P039)         </sp:SignedSupportingTokens>
```

```
(P040)         <sp:EndorsingSupportingTokens>
(P041)           <wsp:Policy>
(P042)             <sp:X509Token sp:IncludeToken=”AlwaysToRecipient">
(P043)               <wsp:Policy>
(P044)                 <sp:WssX509V3Token11/>
(P045)               </wsp:Policy>
(P046)             </sp:X509Token>
(P047)           </wsp:Policy>
(P048)         </sp:EndorsingSupportingTokens>
(P049)         <sp:Wss11>
(P050)           <wsp:Policy>
(P051)             <sp:MustSupportRefKeyIdentifier/>
(P052)             <sp:MustSupportRefIssuerSerial/>
(P053)             <sp:MustSupportRefThumbprint/>
(P054)             <sp:MustSupportRefEncryptedKey/>
(P055)           </wsp:Policy>
(P056)         </sp:Wss11>
(P057)       </wsp:All>
(P058)     </wsp:ExactlyOne>
(P059)   </wsp:Policy>
(P060)
(P061)   <wsp:Policy wsu:Id="SamlForCertificates_input_policy">
(P062)     <wsp:ExactlyOne>
(P063)       <wsp:All>
(P064)         <sp:SignedParts>
(P065)           <sp:Body/>
(P066)         </sp:SignedParts>
(P067)         <sp:EncryptedParts>
(P068)           <sp:Body/>
(P069)         </sp:EncryptedParts>
(P070)       </wsp:All>
(P071)     </wsp:ExactlyOne>
(P072)   </wsp:Policy>
(P073)
(P074)   <wsp:Policy wsu:Id="SamlForCertificate_output_policy">
(P075)     <wsp:ExactlyOne>
(P076)       <wsp:All>
(P077)         <sp:SignedParts>
(P078)           <sp:Body/>
(P079)         </sp:SignedParts>
(P080)         <sp:EncryptedParts>
(P081)           <sp:Body/>
(P082)         </sp:EncryptedParts>
(P083)       </wsp:All>
(P084)     </wsp:ExactlyOne>
(P085)   </wsp:Policy>
```

Lines (P001)-(P059) contain the policy that requires all the contained assertions to be complied with by the Initiator. In this case there are 4 assertions: sp:SymmetricBinding (P004)-(P030), sp:SignedSupportingTokens (P031)-(P039), sp:EndorsingSupportingTokens (P040)-(P048), and sp:Wss11 (P049)-(P056).

The sp:SymmetricBinding assertion (P004) contains an sp:ProtectionToken (P006)-(P016), which indicates that both the Initiator and Recipient must use each other’s public key respectively associated with the X509Token (P008)-(P014) associated with the respective message receiver to encrypt the shared ephemeral symmetric key as explained at the beginning of this section. The messages may either be encrypted and signed using keys derived from the ephemeral key as indicated by the sp:RequireDerivedKeys assertion (P011) or the encryption and signing can be done using the ephemeral key itself without deriving keys, because the RequireDerivedKey assertion is an optional requirement as indicated by the wsp:Optional attribute on line (P011).

The sp:SignedSupportingTokens assertion (P031) contains an sp:SamlToken assertion (P033)-(P037), which indicates that a signed SAML Assertion must always be included in the Initiator’s request to the Recipient (AlwaysToRecipient (P033)). This SAML Assertion must be included in the WS-Security header and referenced and signed as described in the WS-Security 1.1 Profile for SAML [WSS11-SAML1120-PROFILE] as indicated by the sp:WssSamlV11Token11 assertion (P036), which indicates the SAML 1.1 option in that profile (as opposed to the SAML 2.0 option, which would have been indicated by sp:WssSamlV20Token11).

The sp:EndorsingSupportingToken assertion (P040) is needed because there are no guarantees that the ephemeral key, K, is still being used by the Initiator with whom the Recipient would have collaborated originally to establish the ephemeral key as a shared secret. The purpose of the endorsing token is to sign the signature made by the ephemeral key, using the Initiator's private key, which will explicitly guarantee the content of this particular Initiator request. The endorsing token in this policy is an sp:X509Token (P042)-(P046), which, in particular, is an sp:WssX509V3Token11, which must be used in accordance with the WS-Security 1.1 Profile for X509 Tokens [WSS11-X509-PROFILE]. (Note: it may be the case that this X509 certificate is the same X509 certificate referred to in the ProtectionToken assertion (P012). However, it is important to keep in mind that line (P012) only indicates that the Initiator's token will be used by the Recipient to protect the ephemeral key for the symmetric binding. Therefore, the fact that the token is identified in line (P012) as an sp:X509V3Token10 is not directly related to the fact that the same key may be used for the additional purpose of explicitly signing the request message).

The sp:Wss11 assertion (P049-P056) indicates that WS-Security 1.1 constructs are accepted. (Note also that eitherWssX509V3Token10 or WssX509V3Token11 may be used with the Wss11 since both WS-Security 1.0 and WS-Security 1.1 Profiles are supported by WS-Security 1.1)

There are also 2 Policys, one each for the input message and the output message, each of which contains an assertion indicating the message SOAP Body must be signed (P064)-(P066) for the input message and (P077)-(P079) for the output message, and each contains an assertion that the message SOAP Body must be encrypted (P067)-(P069) for the input message and (P080)-(P082) for the output message.

The following is a sample request that is compliant with this policy.

```
(M001)   <?xml version="1.0" encoding="utf-8" ?>
(M002)   <S12:Envelope
(M003)     xmlns:S12="http://schemas.xmlsoap.org/soap/envelope/"
(M004)     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
(M005)     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
(M006)     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
         wss-wssecurity-secext-1.0.xsd"
(M007)     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
         wssecurity-utility-1.0.xsd"
(M008)     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
(M009)     xmlns:xenc="..."
(M010)     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
(M011)    <S12:Header>
(M012)     <wsse:Security S12:mustUnderstand="1">
(M013)      <wsu:Timestamp wsu:Id="timestamp">
(M014)       <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
(M015)      </wsu:Timestamp>
(M016)      <xenc:EncryptedKey Id="encKey" >
(M017)       <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
(M018)        <ds:DigestMethod Algorithm="...#sha1"/>
(M019)       </xenc:EncryptionMethod>
(M020)       <ds:KeyInfo >
(M021)        <wsse:SecurityTokenReference >
(M022)         <wsse:KeyIdentifier EncodingType="...#Base64Binary"
         ValueType="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
(M023)        </wsse:SecurityTokenReference>
(M024)       </ds:KeyInfo>
(M025)       <xenc:CipherData>
(M026)        <xenc:CipherValue>TE...=</xenc:CipherValue>
(M027)       </xenc:CipherData>
(M028)      </xenc:EncryptedKey>
(M029)      <n1:ReferenceList xmlns:n1=".../xmlenc#">
(M030)       <n1:DataReference URI="#encBody"/>
(M031)      </n1:ReferenceList>
```

```
(M032)      <saml:Assertion
(M033)        AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
(M034)        IssueInstant="2003-04-17T00:46:02Z"
(M035)        Issuer="www.opensaml.org"
(M036)        MajorVersion="1"
(M037)        MinorVersion="1">
(M038)       <saml:Conditions
(M039)         NotBefore="2002-06-19T16:53:33.173Z"
(M040)         NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
(M041)       <saml:AttributeStatement>
(M042)        <saml:Subject>
(M043)         <saml:NameIdentifier
(M044)           NameQualifier="www.example.com"
(M045)           Format="">
(M046)          uid=joe,ou=people,ou=saml-demo,o=example.com
(M047)         </saml:NameIdentifier>
(M048)         <saml:SubjectConfirmation>
(M049)          <saml:ConfirmationMethod>
(M050)           urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
(M051)          </saml:ConfirmationMethod>
(M052)         </saml:SubjectConfirmation>
(M053)        </saml:Subject>
(M054)        <saml:Attribute>
(M055)         ...
(M056)        </saml:Attribute>
(M057)        ...
(M058)       </saml:AttributeStatement>
(M059)      </saml:Assertion>
(M060)      <wsse:SecurityTokenReference wsu:id="STR1">
(M061)       <wsse:KeyIdentifier wsu:id="..."
(M062)         ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
         profile-1.0#SAMLAssertionID">
(M063)        _a75adf55-01d7-40cc-929f-dbd8372ebdfc
(M064)       </wsse:KeyIdentifier>
(M065)      </wsse:SecurityTokenReference>
(M066)      <wsse:BinarySecurityToken
(M067)        wsu:Id="attesterCert"
(M068)        ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
         wss-x509-token-profile-1.0#X509v3"
(M069)        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
         200401-wss-soap-message-security-1.0#Base64Binary">
(M070)       MIIEZzCCA9CgAwIBAgIQEmtJZc0...
(M071)      </wsse:BinarySecurityToken>
(M072)      <ds:Signature wsu:Id="message-signature">
(M073)       <ds:SignedInfo>
(M074)        <ds:CanonicalizationMethod
(M075)          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M076)        <ds:SignatureMethod
(M077)          Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
(M078)        <ds:Reference URI="#STR1">
(M079)         <ds:Transforms>
(M080)          <ds:Transform
(M081)            Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-
         200401-wss-soap-message-security-1.0#STR-Transform">
(M082)           <wsse:TransformationParameters>
(M083)            <ds:CanonicalizationMethod
(M084)              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M085)           </wsse:TransformationParameters>
(M086)          </ds:Transform>
(M087)         </ds:Transforms>
(M088)         <ds:DigestMethod
(M089)           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(M090)         <ds:DigestValue>...</ds:DigestValue>
(M091)        </ds:Reference>
```

```
(M092)        <ds:Reference URI="#MsgBody">
(M093)         <ds:DigestMethod
(M094)           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(M095)         <ds:DigestValue>...</ds:DigestValue>
(M096)        </ds:Reference>
(M097)        <ds:Reference URI="#timestamp">
(M098)         <ds:DigestMethod
(M099)           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(M0100)        <ds:DigestValue>...</ds:DigestValue>
(M0101)       </ds:Reference>
(M0102)      </ds:SignedInfo>
(M0103)      <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
(M0104)      <ds:KeyInfo>
(M0105)       <wsse:SecurityTokenReference wsu:id="STR2">
(M0106)        <wsse:Reference URI="#enckey" ValueType="../EncryptedKey"/>
(M0107)       </wsse:SecurityTokenReference>
(M0108)      </ds:KeyInfo>
(M0109)     </ds:Signature>
(M0110)     <ds:Signature wsu:Id="endorsing-signature">
(M0111)      <ds:SignedInfo>
(M0112)       ...
(M0113)       <ds:Reference URI="#message-signature">
(M0114)        ...
(M0115)       </ds:Reference>
(M0116)      </ds:SignedInfo>
(M0117)      <ds:SignatureValue>Bu ...=</ds:SignatureValue>
(M0118)      <ds:KeyInfo>
(M0119)       <wsse:SecurityTokenReference >
(M0120)        <wsse:Reference URI="#attesterCert" ValueType="...#X509v3"/>
(M0121)       </wsse:SecurityTokenReference>
(M0122)      </ds:KeyInfo>
(M0123)     </ds:Signature>
(M0124)    </wsse:Security>
(M0125)   </S12:Header>
(M0126)   <S12:Body wsu:Id="MsgBody">
(M0127)    <xenc:EncryptedData Id="encBody" Type="...#Content"
         MimeType="text/xml" Encoding="UTF-8" >
(M0128)     <xenc:EncryptionMethod Algorithm="http...#aes256-cbc"/>
(M0129)     <ds:KeyInfo >
(M0130)      <wsse:SecurityTokenReference>
(M0131)       <wsse:Reference URI="#enckey" ValueType=".../EncryptedKey"/>
(M0132)      </wsse:SecurityTokenReference>
(M0133)     </ds:KeyInfo>
(M0134)     <xenc:CipherData>
(M0135)      <xenc:CipherValue>70...=</xenc:CipherValue>
(M0136)     </xenc:CipherData>
(M0137)    </xenc:EncryptedData>
(M0138)   </S12:Body>
(M0139)  </S12:Envelope>
```

The message above contains a request compliant with the policy described in (P001)-(P072), which includes the endpoint policy and the input policy.

Lines (M016)-(M028) contain an xenc:EncryptedKey element that contains an Initiator-generated symmetric signing and encryption key, K, that can be used to decrypt the xenc:EncryptedData contained in the SOAP S12:Body (M0126) as indicated by the wsse:SecurityTokenReference (M0130)-(M0132) that uses a direct reference to the Id of the xenc:EncryptedKey, “encKey” on lines (M0131) and (M016).

The xenc:EncryptedKey contains a dsig KeyInfo (M020)-(M024) reference to the Thumbprint of the Recipient’s public X509 certificate (M022). This complies with the policy (P068) that requires the sp:Body to of the input message to be encrypted. It also complies with the policy (P013) to protect using encryption based on X509 token and to refer to it by Thumbprint (P014).

Lines (M029)-(M031) contain an xenc:ReferenceList referring to the xenc:EncryptedData in the S12:Body (M0127)-(M0137).

Lines (M032)-(M059) contain the saml:Assertion as a SignedSupportingToken as required by the endpoint policy (P035). A saml:Assertion used as a SignedSupportingToken uses the "sender-vouches" ConfirmationMethod (M049)-(M051) as described in the introductory section 2.3.

Lines (M060)-(M065) contain a WS-Security wsse:SecurityTokenReference that has a KeyIdentifier of ValueType "…1.0#SAMLAssertionID", which indicates a SAML 1.1 Assertion as described in the WS-Security 1.1 [SAML1120_TOKEN_PROFILE].

Lines (M066)-(M071) contain a wsse:BinarySecurityToken that contains the Initiator's X509 certificate for use as an EndorsingSupportingToken as required by line (P042) of the sp:EndorsingSupportingTokens assertion (P040)-(P048).

Lines (M072)-(M0109) contain the message signature. The dsig Signature covers the following:

- ➢ The SAML Assertion using the ds:Reference (M078)-(M091), which uses the WS-Security STR-Transform technique to refer to the SAML Assertion indirectly through the wsse:SecurityTokenReference (M060) described above. This signature is required by the sp:SignedSupportingTokens assertion (P031)-(P039).
- ➢ The message sp:Body (M0126)-(M0138) using the ds:Reference (M092)-(M096) as required by the input message policy (P065).
- ➢ The message wsu:Timestamp (M013)-(M015) using the ds:Reference (M097)-(M0101) as required by the endpoint policy (P027).

The key used to sign the message signature is referenced in the dsig KeyInfo (M0104)-(M0108), which contains a SecurityTokenReference with a direct URI reference to the xenc:EncryptedKey, "encKey", which contains the Initiator-generated signing and encryption key, K, as described above.

Lines (M0110)-(M0123) contain the endorsing signature. The dsig endorsing Signature covers the following:

- ➢ The message Signature (M072)-(M0109) using the ds:Reference (M0113)-(M0115), which is a direct URI reference to the Id "message-signature" on line (M072).

This signature is required by the policy EndorsingSupportingTokens assertion (P040)-(P048) to be an X509 certificate (P044). The dsig KeyInfo (M0118)-(M0122) contains a WS-Security SecurityTokenReference with a direct URI to the Initiator's X509 certificate on line (M066)-(M071) with Id = "attesterCert".

Lines (M0127)-(M0137) contain the xenc:EncryptedData, which contains the SOAP S12:Body message payload that is encrypted using the encryption key, K, contained in the xenc:EncryptedKey CipherValue (M026), which can only be decrypted using the Recipient's X509 certificate referred to by ThumbprintSHA1 on line (M022).

2.3.2.5 (WSS1.1) SAML1.1/2.0 Holder of Key, Sign, Encrypt

This scenario is based on WS-SX Interop Scenarios Phase 2 (October 31, 2006) [WSSX-WSTR-WSSC-INTEROP] Scenario 5 (Client and STS: Mutual Certificate WSS1.1 (section 3.5 of interop ref), Client and Service: Issued SAML 1.1 Token for Certificate WSS1.1 (section 4.3 of interop ref)).

In this scenario, the service specifies that the client must obtain an IssuedToken from a designated Securtiy Token Service (STS), which must be a SAML 1.1 Assertion. The Assertion contains an ephemeral key K2 in an EncryptedKey element encrypted using service's certificate. The client also obtains the same ephemeral key K2 from the RequestedProofToken returned by the STS. The body of the message from the client to the service is signed using DKT1(K2), encrypted using DKT2(K2), and endorsed using DKT3(K2), which are keys the client derives from K2 using the algorithm specified in section 7 of [WS-SecureConversation]. The response from the service is also signed using derived keys. In a simpler alternative, ephemeral key K itself could be used for message protection.

> Note: In this scenario, in terms of Figure 1 [Figure01], the STS (Issuing Authority) is the Issuer of the SAML 1.1 holder-of-key Assertion that contains the ephemeral symmetric key K2. The service (combined Recipient/RelyingParty) can trust the client (combined

Initiator/Requestor) that uses the ephemeral symmetric key K2 obtained from the RequestedProofToken, because the same key gets delivered to the service in the SAML 1.1 holder-of-key Assertion, which is signed by the trusted Authority.

This scenario is a 2-step sequence from a WS-SP perspective. These 2 steps will be described at a high level first in order to explain the context for the policies and messages that follow.

In step 1 the following takes place:

- the client accesses the service's WS-SP policy (P001 -> P116 below), and determines that it needs to use an IssuedToken from a Security Token Service (STS), specified in the policy. This IssuedToken will serve as the basis of trust by the service. Effectively, the service only trusts the client because the client is able to obtain the IssuedToken from the STS.

To complete step 1, the client will then do the following:

- the client will access the STS and process the STS policy (PSTS-001 -> PSTS-098 below)
- based on the STS policy, the client will send a request to the STS for an IssuedToken (MSTS-001 -> MSTS-0231 below)
- the STS will send a response to the client containing the IssuedToken, which in this example is a SAML 1.1 holder-of-key Assertion (RSTS-001 -> RSTS-0263 below)

In step 2 the following takes place:

- because the client now has the IssuedToken it is now able fulfill the requirements of the service's WS-SP policy (P001-P116 below) that it accessed in step 1
- based on the service's policy the clients sends a request to the service (M001-M0229 below)
- the service will send a response to the client containing the requested resource data (R001-R0153 below)

As an aid to understanding the security properties of the policies and messages in this example, the following is a list of identifiers used in the text descriptions to reference the cryptographic keys that are called for in the policies and used in the messages in this example:

- **X509T1**: Client's (Requestor/Initiator) X509 certificate used for authentication to the STS.
- **X509T2**: STS' (Issuing Authority) X509 certificate used to encrypt keys sent to STS, used to authenticate STS to Service.
- **X509T3**: Service's (Recipient/RelyingParty/ValidatingAuthority) X509 certificate used to encrypt keys sent to Service.
- **K1**: Client-generated ephemeral symmetric key for crypto communication to the STS.
- **K2**: Client-generated ephemeral symmetric key for crypto communication between the Client and the Service.
- **K3**: STS-generated proof key for use by Client to authenticate with Service via saml:Assertion.
- **DKT1(K2)**: Client-generated derived key used for signing requests to the Service.
- **DKT2(K2)**: Client-generated derived key used for encrypting requests to the Service.
- **DKT3(K3)**: Client-generated derived key used for endorsing signed requests to the Service.
- **DKT4(K2)**: Service-generated derived key used for encrypting responses to the Client.
- **DKT5(K2)**: Service-generated derived key used for signing responses to the Client.

Here is the WS-SP Policy that the Service presents to a Client:

```
(P001)     <wsp:Policy wsu:Id="Service5-Policy"
(P002)
(P003)       xmlns:wsp="http://www.w3.org/ns/ws-policy"
(P004)       xmlns:sp"..."
(P005)
```

```
(P006)       xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
           wssecurity-utility-1.0.xsd"
(P007)     >
(P008)       <wsp:ExactlyOne>
(P009)         <wsp:All>
(P010)           <sp:SymmetricBinding>
(P011)             <wsp:Policy>
(P012)               <sp:ProtectionToken>
(P013)                 <wsp:Policy>
(P014)                   <sp:X509Token IncludeToken=
(P015)     "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken
           /Never">
(P016)                     <wsp:Policy>
(P017)                       <sp:RequireDerivedKeys/>
(P018)                       <sp:RequireThumbprintReference/>
(P019)                       <sp:WssX509V3Token10/>
(P020)                     </wsp:Policy>
(P021)                   </sp:X509Token>
(P022)                 </wsp:Policy>
(P023)               </sp:ProtectionToken>
(P024)               <sp:AlgorithmSuite>
(P025)                 <wsp:Policy>
(P026)                   <sp:Basic256/>
(P027)                 </wsp:Policy>
(P028)               </sp:AlgorithmSuite>
(P029)               <sp:Layout>
(P030)                 <wsp:Policy>
(P031)                   <sp:Strict/>
(P032)                 </wsp:Policy>
(P033)               </sp:Layout>
(P034)               <sp:IncludeTimestamp/>
(P035)               <sp:OnlySignEntireHeadersAndBody/>
(P036)             </wsp:Policy>
(P037)           </sp:SymmetricBinding>
(P038)           <sp:EndorsingSupportingTokens>
(P039)             <wsp:Policy>
(P040)               <sp:IssuedToken IncludeToken=
(P041)     "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken
           /AlwaysToRecipient">
(P042)                 <sp:Issuer>
(P043)                   <a:Address>http://example.com/STS</a:Address>
(P044)                 </sp:Issuer>
(P045)                 <sp:RequestSecurityTokenTemplate>
(P046)                   <t:TokenType
(P047)     >http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
           1.1#SAMLV1.1</t:TokenType>
(P048)                   <t:KeyType
(P049)     >http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey</t:KeyTy
           pe>
(P050)                   <t:KeySize>256</t:KeySize>
(P051)                   <t:CanonicalizationAlgorithm
(P052)     >http://www.w3.org/2001/10/xml-exc-c14n#</t:CanonicalizationAlgorithm>
(P053)                   <t:EncryptionAlgorithm
(P054)     >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptionAlgorithm>
(P055)                   <t:EncryptWith
(P056)     >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptWith>
(P057)                   <t:SignWith
(P058)     >http://www.w3.org/2000/09/xmldsig#hmac-sha1</t:SignWith>
(P059)                 </sp:RequestSecurityTokenTemplate>
(P060)                 <wsp:Policy>
(P061)                   <sp:RequireDerivedKeys/>
(P062)                   <sp:RequireInternalReference/>
(P063)                 </wsp:Policy>
(P064)               </sp:IssuedToken>
```

```
(P065)             </wsp:Policy>
(P066)           </sp:EndorsingSupportingTokens>
(P067)           <sp:Wss11>
(P068)             <wsp:Policy>
(P069)               <sp:MustSupportRefKeyIdentifier/>
(P070)               <sp:MustSupportRefIssuerSerial/>
(P071)               <sp:MustSupportRefThumbprint/>
(P072)               <sp:MustSupportRefEncryptedKey/>
(P073)               <sp:RequireSignatureConfirmation/>
(P074)             </wsp:Policy>
(P075)           </sp:Wss11>
(P076)           <sp:Trust13>
(P077)             <wsp:Policy>
(P078)               <sp:MustSupportIssuedTokens/>
(P079)               <sp:RequireClientEntropy/>
(P080)               <sp:RequireServerEntropy/>
(P081)             </wsp:Policy>
(P082)           </sp:Trust13>
(P083)         </wsp:All>
(P084)       </wsp:ExactlyOne>
(P085)     </wsp:Policy>

(P086)     <wsp:Policy wsu:Id="InOut-Policy"
(P087)
(P088)       xmlns:wsp="http://www.w3.org/ns/ws-policy"
(P089)       xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
(P090)
(P091)       xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
           wssecurity-utility-1.0.xsd">
(P092)       <wsp:ExactlyOne>
(P093)         <wsp:All>
(P094)           <sp:SignedParts>
(P095)             <sp:Body/>
(P096)             <sp:Header Name="To"
(P097)               Namespace="http://www.w3.org/2005/08/addressing"/>
(P098)             <sp:Header Name="From"
(P099)               Namespace="http://www.w3.org/2005/08/addressing"/>
(P0100)            <sp:Header Name="FaultTo"
(P0101)              Namespace="http://www.w3.org/2005/08/addressing"/>
(P0102)            <sp:Header Name="ReplyTo"
(P0103)              Namespace="http://www.w3.org/2005/08/addressing"/>
(P0104)            <sp:Header Name="MessageID"
(P0105)              Namespace="http://www.w3.org/2005/08/addressing"/>
(P0106)            <sp:Header Name="RelatesTo"
(P0107)              Namespace="http://www.w3.org/2005/08/addressing"/>
(P0108)            <sp:Header Name="Action"
(P0109)              Namespace="http://www.w3.org/2005/08/addressing"/>
(P0110)          </sp:SignedParts>
(P0111)          <sp:EncryptedParts>
(P0112)            <sp:Body/>
(P0113)          </sp:EncryptedParts>
(P0114)        </wsp:All>
(P0115)      </wsp:ExactlyOne>
(P0116)    </wsp:Policy>
```

When a Client encounters the above Service Policy, it determines that it must obtain an IssuedToken from the designated Issuer. After obtaining the IssuedToken, the client may send the request in accordance with the rest of the policy. Details of the Service Policy follow. The STS Policy its details follow after the Service Policy.

Lines (P001)-(P085) above contain the Service Endpoint wsp:Policy, which contains a wsp:All assertion that requires all 4 of its contained assertions to be complied with: sp:SymmetricBinding (P010)-(P037), sp:EndorsingSupportingTokens (P038)-(P066), sp:Wss11 (P067)-(P075), and sp:Trust13 (P076)-(P082).

Lines (P010)-(P037) contain the **sp:SymmetricBinding assertion** that requires that derived keys (DKT1, DKT2, DKT3) be used to protect the message (P017) and that the ephemeral key (K2) used to derive these keys be encrypted using the service's X509 certificate (X509T3) as specified by the sp:X509Token assertion (P014)-(P021), which also indicates that the X509 token, itself should not be sent (P015), but that a Thumbprint reference (P018) to it be sent. Finally, the sp:SymmetricBinding specifies that the sp:Basic256 sp:AlgorithmSuite be used (P024)-(P028), that sp:Strict sp:Layout be used (P029)-(P033), that an sp:Timestamp be included (P034), and that only the complete message Body and Headers be signed (P035), where the Headers part means that only direct child elements of the WS-Security SOAP header element be signed.

Lines (P038)-(P066) contain the **sp:EndorsingSupportingTokens assertion** that an sp:IssuedToken (P040)-(P064) be used to sign the message signature and that the IssuedToken must be included with the request (P040)-(P041). Lines (P042)-(P044) specify the address of the SecurityTokenService (STS) from which the IssuedToken must be obtained. Lines (P045)-(P059) contain an sp:RequestSecurityTokenTemplate (P046)-(P058) which contains explicit WS-Trust elements that the client should directly copy to a t:SecondaryParameters element to include with the WS-Trust t:RequestSecurityToken to the STS to obtain the sp:IssuedToken. Of particular interest here is that the t:TokenType (P046)-(P047) requested is a SAML 1.1 Assertion, which will be used to contain the ephemeral symmetric key (K2) (P048)-(P049). K2 will be used for communication between the Client and the Service. K2 will be encrypted by the STS using the Service's X509 certificate (X509T3). The Client is also informed by the IssuedToken assertion that the IssuedToken may only be referenced internally within the message (P062) and that the ephemeral key (K2) associated with the IssuedToken be used by the Client to derive the keys (DKT1(K2), DKT2(K2), DKT3(K2)) used in the Client's request to the Service.

Lines (P067)-(P075) contain the **sp:Wss11 assertion** that indicates that WS-Security 1.1 will be used, which includes Wss11-only features such as Thumbprint (P073), EncryptedKey (P074), and SignatureConfirmation (P075).

Lines (P076)-(P082) contain the **sp:Trust13 assertion** that indicates the Client should expect to use WS-Trust 1.3 (WSTRUST) to obtain the IssuedToken from the STS.

Lines (P086)-(P0116) contain the **operation input and output policies** that the client should use to determine what parts of the messages are to be signed (P094)-(P0110) and encrypted (P0111)-(P0113).

Here is the WS-SP Policy that the STS presents to a Client:

```
(PSTS-001)    <wsp:Policy wsu:Id="STS5-Policy"
(PSTS-002)
(PSTS-003)      xmlns:wsp="http://www.w3.org/ns/ws-policy"
(PSTS-004)    xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
(PSTS-005)
(PSTS-006)      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
              wss-wssecurity-utility-1.0.xsd"
(PSTS-007)    >
(PSTS-008)      <wsp:ExactlyOne>
(PSTS-009)        <wsp:All>
(PSTS-010)          <sp:SymmetricBinding>
(PSTS-011)            <wsp:Policy>
(PSTS-012)              <sp:ProtectionToken>
(PSTS-013)                <wsp:Policy>
(PSTS-014)                  <sp:X509Token IncludeToken=
(PSTS-015)    "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeTo
              ken/Never">
(PSTS-016)                    <wsp:Policy>
(PSTS-017)                      <sp:RequireThumbprintReference/>
(PSTS-018)                      <sp:WssX509V3Token10/>
(PSTS-019)                    </wsp:Policy>
(PSTS-020)                  </sp:X509Token>
(PSTS-021)                </wsp:Policy>
(PSTS-022)              </sp:ProtectionToken>
(PSTS-023)              <sp:AlgorithmSuite>
(PSTS-024)                <wsp:Policy>
(PSTS-025)                  <sp:Basic256/>
```

```
(PSTS-026)                </wsp:Policy>
(PSTS-027)              </sp:AlgorithmSuite>
(PSTS-028)              <sp:Layout>
(PSTS-029)                <wsp:Policy>
(PSTS-030)                  <sp:Strict/>
(PSTS-031)                </wsp:Policy>
(PSTS-032)              </sp:Layout>
(PSTS-033)              <sp:IncludeTimestamp/>
(PSTS-034)              <sp:OnlySignEntireHeadersAndBody/>
(PSTS-035)            </wsp:Policy>
(PSTS-036)          </sp:SymmetricBinding>
(PSTS-037)          <sp:EndorsingSupportingTokens>
(PSTS-038)            <wsp:Policy>
(PSTS-039)              <sp:X509Token IncludeToken=
(PSTS-040)    "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeTo
              ken/AlwaysToRecipient">
(PSTS-041)                <wsp:Policy>
(PSTS-042)                  <sp:RequireThumbprintReference/>
(PSTS-043)                  <sp:WssX509V3Token10/>
(PSTS-044)                </wsp:Policy>
(PSTS-045)              </sp:X509Token>
(PSTS-046)            </wsp:Policy>
(PSTS-047)          </sp:EndorsingSupportingTokens>
(PSTS-048)          <sp:Wss11>
(PSTS-049)            <wsp:Policy>
(PSTS-050)              <sp:MustSupportRefKeyIdentifier/>
(PSTS-051)              <sp:MustSupportRefIssuerSerial/>
(PSTS-052)              <sp:MustSupportRefThumbprint/>
(PSTS-053)              <sp:MustSupportRefEncryptedKey/>
(PSTS-054)              <sp:RequireSignatureConfirmation/>
(PSTS-055)            </wsp:Policy>
(PSTS-056)          </sp:Wss11>
(PSTS-057)          <sp:Trust13>
(PSTS-058)            <wsp:Policy>
(PSTS-059)              <sp:MustSupportIssuedTokens/>
(PSTS-060)              <sp:RequireClientEntropy/>
(PSTS-061)              <sp:RequireServerEntropy/>
(PSTS-062)            </wsp:Policy>
(PSTS-063)          </sp:Trust13>
(PSTS-064)        </wsp:All>
(PSTS-065)      </wsp:ExactlyOne>
(PSTS-066)    </wsp:Policy>

(PSTS-067)    <wsp:Policy wsu:Id="InOut-Policy"
(PSTS-068)
(PSTS-069)      xmlns:wsp="http://www.w3.org/ns/ws-policy"
(PSTS-070)    xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
(PSTS-071)
(PSTS-072)      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
              wss-wssecurity-utility-1.0.xsd"
(PSTS-073)    >
(PSTS-074)      <wsp:ExactlyOne>
(PSTS-075)        <wsp:All>
(PSTS-076)          <sp:SignedParts>
(PSTS-077)            <sp:Body/>
(PSTS-078)            <sp:Header Name="To"
(PSTS-079)              Namespace="http://www.w3.org/2005/08/addressing"/>
(PSTS-080)            <sp:Header Name="From"
(PSTS-081)              Namespace="http://www.w3.org/2005/08/addressing"/>
(PSTS-082)            <sp:Header Name="FaultTo"
(PSTS-083)              Namespace="http://www.w3.org/2005/08/addressing"/>
(PSTS-084)            <sp:Header Name="ReplyTo"
(PSTS-085)              Namespace="http://www.w3.org/2005/08/addressing"/>
(PSTS-086)            <sp:Header Name="MessageID"
```

```
(PSTS-087)              Namespace="http://www.w3.org/2005/08/addressing"/>
(PSTS-088)            <sp:Header Name="RelatesTo"
(PSTS-089)              Namespace="http://www.w3.org/2005/08/addressing"/>
(PSTS-090)            <sp:Header Name="Action"
(PSTS-091)              Namespace="http://www.w3.org/2005/08/addressing"/>
(PSTS-092)          </sp:SignedParts>
(PSTS-093)          <sp:EncryptedParts>
(PSTS-094)            <sp:Body/>
(PSTS-095)          </sp:EncryptedParts>
(PSTS-096)        </wsp:All>
(PSTS-097)      </wsp:ExactlyOne>
(PSTS-098)    </wsp:Policy>
```

Above is the STS Policy that the Client will encounter when obtaining the IssuedToken required by the Service Policy. This policy is quite similar in detail to the Service Policy and therefore only the differences that are noteworthy will be discussed in the details below.

Similar to the Service Policy, the STS endpoint Policy contains 4 assertions to be complied with: sp:SymmetricBinding (PSTS-010)-(PSTS-036), sp:EndorsingSupportingTokens (PSTS-037)-(PSTS-047), sp:Wss11 (PSTS-048)-(PSTS-056), and sp:Trust13 (PSTS-057)-(PSTS-063).

Lines (PSTS-010)-(PSTS-036) contain the **sp:SymmetricBinding assertion**, where the main difference from the previous policy is that here the sp:ProtectionToken is an sp:X509Token that will be used to encrypt the ephemeral client-generated key (K1) that will be used for Client-STS communication. Derived keys will not be required in this communication.

Lines (PSTS-037)-(PSTS-047) contain the **sp:EndorsingSupportingTokens assertion**, which in this case contains an sp:X509Token assertion (PSTS-039)-(PSTS-045) that requires the Client to include (PSTS-040) its X509 certificate, which must be used to sign the message signature. The STS uses this mechanism to authenticate the Client.

The **sp:Wss11 assertion** (PSTS-048)-(PSTS-056), **sp:Trust13 assertion** (PSTS-057)-(PSTS-063) and the **operation input and output policies** (PSTS-067)-(PSTS-098) are the same as those described for the Service policy above (P067)-(P0116).

Below are included sample messages that comply with the above policies. The messages are presented in the same order that they would be used in a real scenario and therefore the Client-STS request (MSTS-001)-(MSTS-0231) and response (RSTS-001)-(RSTS-0263) are presented first, which are then followed by the Client-Service request (M001-M229) and response (R001)-(R153).

Here is an example Client request to the STS:

```
(MSTS-001)    <s:Envelope xmlns:s=http://schemas.xmlsoap.org/soap/envelope
(MSTS-002)      xmlns:a=http://www.w3.org/2005/08/addressing
(MSTS-003)      xmlns:e=http://www.w3.org/2001/04/xmlenc#
(MSTS-004)      xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
              wssecurity-secext-1.0.xsd"
(MSTS-005)      xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
              wssecurity-utility-1.0.xsd"  >
(MSTS-006)      <s:Header>
(MSTS-007)        <a:Action s:mustUnderstand="1" u:Id="_3">
(MSTS-008)          http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue
(MSTS-009)        </a:Action>
(MSTS-010)        <a:MessageID u:Id="_4">
(MSTS-011)          urn:uuid:04d386bf-f850-459e-918b-ad80f3d1e088
(MSTS-012)        </a:MessageID>
(MSTS-013)        <a:ReplyTo u:Id="_5">
(MSTS-014)          <a:Address>
(MSTS-015)            http://www.w3.org/2005/08/addressing/anonymous
(MSTS-016)          </a:Address>
(MSTS-017)        </a:ReplyTo>
(MSTS-018)        <a:To s:mustUnderstand="1" u:Id="_6">
(MSTS-019)          http://server.example.com/STS/Scenarios5-6
(MSTS-020)        </a:To>
(MSTS-021)        <o:Security s:mustUnderstand="1">
```

```
(MSTS-022)          <u:Timestamp
(MSTS-023)              u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-10">
(MSTS-024)            <u:Created>2005-10-25T00:47:36.144Z</u:Created>
(MSTS-025)            <u:Expires>2005-10-25T00:52:36.144Z</u:Expires>
(MSTS-026)          </u:Timestamp>
(MSTS-027)          <e:EncryptedKey
(MSTS-028)              Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-9">
(MSTS-029)            <e:EncryptionMethod Algorithm=
(MSTS-030)                 "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
(MSTS-031)            <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
(MSTS-032)              <o:SecurityTokenReference>
(MSTS-033)                <o:KeyIdentifier ValueType=
(MSTS-034)    "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
              1.1.xsd#ThumbprintSHA1">
(MSTS-035)                  W+rgYBmLmVEG//scD7Vo8Kq5G7I=
(MSTS-036)                </o:KeyIdentifier>
(MSTS-037)              </o:SecurityTokenReference>
(MSTS-038)            </KeyInfo>
(MSTS-039)            <e:CipherData>
(MSTS-040)              <e:CipherValue>
(MSTS-041)                <!--base64 encoded cipher-->
(MSTS-042)              </e:CipherValue>
(MSTS-043)            </e:CipherData>
(MSTS-044)            <e:ReferenceList>
(MSTS-045)              <e:DataReference URI="#_2"/>
(MSTS-046)            </e:ReferenceList>
(MSTS-047)          </e:EncryptedKey>
(MSTS-048)          <o:BinarySecurityToken
(MSTS-049)            u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-6"
(MSTS-050)            ValueType=
(MSTS-051)    "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
              profile-1.0#X509v3"
(MSTS-052)            EncodingType=
(MSTS-053)    "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
              message-security-1.0#Base64Binary">
(MSTS-054)
(MSTS-055)    MIIDDDCCAfSgAwIBAgIQM6YEf7FVYx/tZyEXgVComTANBgkqhkiG9w0BAQUFADAwMQ4w
(MSTS-056)    DAYDVQQKDAVPQVNJUzEeMBwGA1UEAwwVT0FTSVMgSW50ZXJvcCBUZXN0IENBMB4XDTA1
(MSTS-057)    MDMxOTAwMDAwMFoXDTE4MDMxOTIzNTk1OVowQjEOMAwGA1UECgwFT0FTSVMxIDAeBgNV
(MSTS-058)    BAsMF09BU0lTIEludGVyb3AgVGVzdCBDZXJ0MQ4wDAYDVQQDDAVBbGljZTCBnzANBgkq
(MSTS-059)    hkiG9w0BAQEFAAOBjQAwgYkCgYEAoqi99By1VYo0aHrkKCNT4DkIgPL/SgahbeKdGhrb
(MSTS-060)    u3K2XG7arfD9tqIBIKMfrX4Gp90NJa85AV1yiNsEyvq+mUnMpNcKnLXLOjkTmMCqDYbb
(MSTS-061)    kehJlXPnaWLzve+mW0pJdPxtf3rbD4PS/cBQIvtpjmrDAU8VsZKT8DN5Kyz+EzsCAwEA
(MSTS-062)    AaOBkzCBkDAJBgNVHRMEAjAAMDMGA1UdHwQsMCowKKImhiRodHRwOi8vaW50ZXJvcC5i
(MSTS-063)    YnRlc3QubmV0L2NybC9jYS5jcmwwDgYDVR0PAQH/BAQDAgSwMB0GA1UdDgQWBBQK4l0T
(MSTS-064)    UHZ1QV3V2QtlLNDm+PoxiDAfBgNVHSMEGDAWgBTAnSj8wes1oR3WqqqgHBpNwkkPDzAN
(MSTS-065)    BgkqhkiG9w0BAQUFAAOCAQEABTqpOpvW+6yrLXyUlP2xJbEkohXHI5OWwKWleOb9hlkh
(MSTS-066)    WntUalfcFOJAgUyH30TTpHldzx1+vK2LPzhoUFKYHE1IyQvokBN2JjFO64BqukCKnZhl
(MSTS-067)    dLRPxGhfkTdxQgdf5rCK/wh3xVsZCNTfuMNmlAM6lOAg8QduDah3WFZpEA0s2nwQaCNQ
(MSTS-068)    TNMjJC8tav1CBr6+E5FAmwPXP7pJxn9Fw9OXRyqbRA4v2y7YpbGkG2GI9UvOHw6SGvf4
(MSTS-069)    FRSthMMO35YbpikGsLix3vAsXWWi4rwfVOYzQK0OFPNi9RMCUdSH06m9uLWckiCxjos0
(MSTS-070)    FQODZE9l4ATGy9s9hNVwryOJTw==
(MSTS-071)          </o:BinarySecurityToken>
(MSTS-072)          <Signature Id="_0" xmlns="http://www.w3.org/2000/09/xmldsig#">
(MSTS-073)            <SignedInfo>
(MSTS-074)              <CanonicalizationMethod Algorithm=
(MSTS-075)                  "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(MSTS-076)              <SignatureMethod Algorithm=
(MSTS-077)                  "http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
(MSTS-078)              <Reference URI="#_1">
(MSTS-079)                <Transforms>
(MSTS-080)                  <Transform Algorithm=
(MSTS-081)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(MSTS-082)                </Transforms>
```

```
(MSTS-083)                 <DigestMethod Algorithm=
(MSTS-084)                     "http://www.w3.org/2000/09/xmldsig#sha1"/>
(MSTS-085)                 <DigestValue>VX1fCPwCzVsSc1hZf0BSbCgW2hM=</DigestValue>
(MSTS-086)               </Reference>
(MSTS-087)               <Reference URI="#_3">
(MSTS-088)                 <Transforms>
(MSTS-089)                   <Transform Algorithm=
(MSTS-090)                       "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(MSTS-091)                 </Transforms>
(MSTS-092)                 <DigestMethod Algorithm=
(MSTS-093)                     "http://www.w3.org/2000/09/xmldsig#sha1"/>
(MSTS-094)                 <DigestValue>FwiFAUuqNDo9SDkk5A28Mg7Pa8Q=</DigestValue>
(MSTS-095)               </Reference>
(MSTS-096)               <Reference URI="#_4">
(MSTS-097)                 <Transforms>
(MSTS-098)                   <Transform Algorithm=
(MSTS-099)                       "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(MSTS-0100)                </Transforms>
(MSTS-0101)                <DigestMethod Algorithm=
(MSTS-0102)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(MSTS-0103)                <DigestValue>oM59PsOTpmrDdOcwXYQzjVUl0xw=</DigestValue>
(MSTS-0104)              </Reference>
(MSTS-0105)              <Reference URI="#_5">
(MSTS-0106)                <Transforms>
(MSTS-0107)                  <Transform Algorithm=
(MSTS-0108)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(MSTS-0109)                </Transforms>
(MSTS-0110)                <DigestMethod Algorithm=
(MSTS-0111)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(MSTS-0112)                <DigestValue>KIK3vklFN1QmMdQkplq2azfzrzg=</DigestValue>
(MSTS-0113)              </Reference>
(MSTS-0114)              <Reference URI="#_6">
(MSTS-0115)                <Transforms>
(MSTS-0116)                  <Transform Algorithm=
(MSTS-0117)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(MSTS-0118)                </Transforms>
(MSTS-0119)                <DigestMethod Algorithm=
(MSTS-0120)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(MSTS-0121)                <DigestValue>RJEe3hrcyCD6PzFJo6fyut6biVg=</DigestValue>
(MSTS-0122)              </Reference>
(MSTS-0123)              <Reference
(MSTS-0124)                  URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-10">
(MSTS-0125)                <Transforms>
(MSTS-0126)                  <Transform Algorithm=
(MSTS-0127)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(MSTS-0128)                </Transforms>
(MSTS-0129)                <DigestMethod Algorithm=
(MSTS-0130)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(MSTS-0131)                <DigestValue>zQdN5XpejfqXn0WKo0m51ZYiasE=</DigestValue>
(MSTS-0132)              </Reference>
(MSTS-0133)            </SignedInfo>
(MSTS-0134)            <SignatureValue
(MSTS-0135)                >iHGJ+xV2VZTjMlRc7AQJrwLY/aM=</SignatureValue>
(MSTS-0136)            <KeyInfo>
(MSTS-0137)              <o:SecurityTokenReference>
(MSTS-0138)                <o:Reference
(MSTS-0139)                  ValueType=
(MSTS-0140)   "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
              1.1.xsd#EncryptedKey"
(MSTS-0141)                  URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-9"/>
(MSTS-0142)              </o:SecurityTokenReference>
(MSTS-0143)            </KeyInfo>
(MSTS-0144)          </Signature>
(MSTS-0145)          <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
```

```
(MSTS-0146)           <SignedInfo>
(MSTS-0147)             <CanonicalizationMethod Algorithm=
(MSTS-0148)                 "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(MSTS-0149)             <SignatureMethod Algorithm=
(MSTS-0150)                  "http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
(MSTS-0151)             <Reference URI="#_0">
(MSTS-0152)               <Transforms>
(MSTS-0153)                 <Transform Algorithm=
(MSTS-0154)                     "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(MSTS-0155)               </Transforms>
(MSTS-0156)               <DigestMethod Algorithm=
(MSTS-0157)                   "http://www.w3.org/2000/09/xmldsig#sha1"/>
(MSTS-0158)               <DigestValue>UZKtShk8q6iu9WR5uQZp04iAitg=</DigestValue>
(MSTS-0159)             </Reference>
(MSTS-0160)           </SignedInfo>
(MSTS-0161)           <SignatureValue>
(MSTS-0162)   Ovxdeg4KQcfQlT/hEBJz+Z8dQUAfChaWIcmG3xGLZYcc8tbmCtZFuQz9tnW35Lmst6vI
(MSTS-0163)   RefuPA7ewRLYORAOjf92SxMbeVTlrIxQbIQNw0bs4SBSLfAo14=
(MSTS-0164)           </SignatureValue>
(MSTS-0165)           <KeyInfo>
(MSTS-0166)             <o:SecurityTokenReference>
(MSTS-0167)               <o:Reference
(MSTS-0168)                 ValueType=
(MSTS-0169)   "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
              profile-1.0#X509v3"
(MSTS-0170)                 URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-6"/>
(MSTS-0171)             </o:SecurityTokenReference>
(MSTS-0172)           </KeyInfo>
(MSTS-0173)         </Signature>
(MSTS-0174)       </o:Security>
(MSTS-0175)     </s:Header>
(MSTS-0176)     <s:Body u:Id="_1">
(MSTS-0177)       <e:EncryptedData
(MSTS-0178)           Id="_2"
(MSTS-0179)           Type="http://www.w3.org/2001/04/xmlenc#Content">
(MSTS-0180)         <e:EncryptionMethod Algorithm=
(MSTS-0181)             "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
(MSTS-0182)         <e:CipherData>
(MSTS-0183)           <e:CipherValue>
(MSTS-0184)             <!-- base64 encoded octets with encrypted RST request-->
(MSTS-0185)             <!-- Unencrypted form:  -->
(MSTS-0186)             <!—
(MSTS-0187)       <t:RequestSecurityToken>
(MSTS-0188)         <t:RequestType>
(MSTS-0189)           http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
(MSTS-0190)         </t:RequestType>
(MSTS-0191)         <wsp:AppliesTo
(MSTS-0192)             xmlns:wsp="http://www.w3.org/ns/ws-policy">
(MSTS-0193)           <a:EndpointReference
(MSTS-0194)               xmlns:a="http://www.w3.org/2005/08/addressing">
(MSTS-0195)             <a:Address
(MSTS-0196)               >http://server.example.com/Scenarios5</a:Address>
(MSTS-0197)           </a:EndpointReference>
(MSTS-0198)         </wsp:AppliesTo>
(MSTS-0199)         <t:Entropy>
(MSTS-0200)           <t:BinarySecret
(MSTS-0201)             u:Id="uuid-4acf589c-0076-4a83-8b66-5f29341514b7-3"
(MSTS-0202)             Type=
(MSTS-0203)               "http://docs.oasis-open.org/ws-sx/ws-trust/200512/Nonce"
(MSTS-0204)         >Uv38QLxDQM9gLoDZ6OwYDiFk094nmwu3Wmay7EdKmhw=</t:BinarySecret>
(MSTS-0205)         </t:Entropy>
(MSTS-0206)         <t:KeyType>
(MSTS-0207)          http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey
(MSTS-0208)         </t:KeyType>
```

```
(MSTS-0209)         <t:KeySize>256</t:KeySize>
(MSTS-0210)         <t:ComputedKeyAlgorithm>
(MSTS-0211)           http://docs.oasis-open.org/ws-sx/ws-trust/200512/CK/PSHA1
(MSTS-0212)         </t:ComputedKeyAlgorithm>
(MSTS-0213)         <t:SecondaryParameters>
(MSTS-0214)            <t:TokenType
(MSTS-0215)   >http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
              1.1#SAMLV1.1</t:TokenType>
(MSTS-0216)            <t:KeyType
(MSTS-0217)   >http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey</t:Ke
              yType>
(MSTS-0218)            <t:KeySize>256</t:KeySize>
(MSTS-0219)            <t:CanonicalizationAlgorithm
(MSTS-0220)   >http://www.w3.org/2001/10/xml-exc-c14n#</t:CanonicalizationAlgorith
              m>
(MSTS-0221)            <t:EncryptionAlgorithm
              >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptionAlgorithm>
(MSTS-0222)            <t:EncryptWith
              >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptWith>
(MSTS-0223)            <t:SignWith
              >http://www.w3.org/2000/09/xmldsig#hmac-sha1</t:SignWith>
(MSTS-0224)         </t:SecondaryParameters>
(MSTS-0225)       </t:RequestSecurityToken>
(MSTS-0226)                      -->
(MSTS-0227)           </e:CipherValue>
(MSTS-0228)         </e:CipherData>
(MSTS-0229)       </e:EncryptedData>
(MSTS-0230)     </s:Body>
(MSTS-0231)   </s:Envelope>
```

The message above is a request by the Client to the STS in compliance with the STS policy (PSTS-001)-(PSTS-098). Salient features of this message appropriate to the compliance are described below.

Lines (MSTS-027)-(MSTS-047) contain the **e:EncryptedKey element**, which contains the encrypted client-generated ephemeral key (K1) (MSTS-039)-(MSTS-043). K1 is encrypted using the STS certificate (X509T2) which is specified by its Thumbprint identifier in the WS-Security o:SecurityTokenRefernece (MSTS-032)-(MSTS-037). The e:ReferenceList in the e:EncryptedKey (MSTS-044)-(MSTS-046) indicates that the encryption key K1 is used to directly encrypt the message Body which is referenced by the e:DataReference URI="#_2" (MSTS-045).

Lines (MSTS-048)-(MSTS-071) contain a **WS-Security o:BinarySecurityToken**, which contains the Client's public X509 certificate (X509T1) that is required for Client authentication to the STS by the sp:EndorsingToken in the STS policy (PSTS-037)-(PSTS-047).

Lines (MSTS-072)-(MSTS-0144) contain the **WS-SP message signature** in an XML Digital Signature (dsig) element (namespace = …xmldsig (MSTS-072)). The elements covered by the dsig Signature are identified in the dsig Reference elements:

- dsig Reference (MSTS-078) covers the s:Body (Id="_1" (MSTS-0176))
- dsig Reference (MSTS-087) covers the a:Action (Id="_3" (MSTS-007))
- dsig Reference (MSTS-096) covers the a:MessageID (Id="_4" (MSTS-010))
- dsig Reference (MSTS-0105) covers the a:ReplyTo (Id="_5" (MSTS-013))
- dsig Reference (MSTS-0114) covers the a:To (Id="_6" (MSTS-018))
- dsig Reference (MSTS-0123) covers the u:Timestamp (Id="uuid … 3849-10" (MSTS-023))

all as required by the STS Input and Output Policy sp:SignedParts (PSTS-076)-(PSTS-092), which covers the first 5 elements above (note: if an element is not present that is identified in the policy (such as FaultTo or RelatesTo) it obviously is not required to be signed, but if present must be signed). The u:Timestamp is required to be signed by its presence in the STS endpoint policy (PSTS-033).

Lines (MSTS-0136)-(MSTS-0143) of the **WS-SP message signature contain the dsig KeyInfo**, which contains a WS-Security o:SecurityTokenReference, which contains an o:Reference to the signing validation key (i.e. the key which can be used to verify this dsig:Signature), which in this case is contained in the e:EncryptedKey element (Id=”uuid … 3849-9” (MSTS-027)) that was described above. Note: at this point to verify the Signature, the STS will decrypt the e:EncryptedKey using the private key from the STS certificate, X509T2, which will produce the client-generated ephemeral signing and encryption key, K1, which, in turn, may be used to validate the message signature. Note also: at this point the STS only knows whether the data covered by the message signature is valid or not, but the STS does not yet know the identity of the entity that actually sent the data. That is covered next:

Lines (MSTS-0145)-(MSTS-0173) contain the **WS-SP endorsing signature**, also in an XML Digital Signature element. The endorsing signature only covers one element, the message signature element, which is identified by the dsig Reference element:

- dsig Reference (MSTS-0151) covers the dsig Signature (Id=”_0” (MSTS-072))

as required by the STS endpoint policy sp:EndorsingSupportingTokens (PSTS-037)-(PSTS-047).

Lines (MSTS-0165)-(MSTS-0174) of the **WS-SP endorsing signature contain the dsig KeyInfo**, which contains a WS-Security o:SecurityTokenReference, which contains an o:Reference to the signing validation key, which in this case is in the o:BinarySecurityToken element (Id=”uuid … 3849-6” (MSTS-048)) that was also described above. Note: now the STS finally has the credentials necessary to authenticate the Client. The STS will generally have an identity database of trusted clients and their associated X509 certificates. If a client successfully produces a signature that can be validated by the associated certificate in the STS database, which would have to match the certificate in the o:BinarySecurityToken element, then the client is presumed to be authenticated to the level of security provided by this mechanism (strong single factor authentication).

Lines (MSTS-0176)-(MSTS-0230) contain the **SOAP message s:Body element**, which contains its payload in an e:EncryptedData element (MSTS-0177)-(MSTS-0229). This e:EncryptedData element was identified above by its Id=”_2” in the e:ReferenceList (MSTS-044) of the e:EncryptedKey that was processed by the STS above to obtain the client ephemeral key (K1), with which this e:EncryptedData can be decrypted. Generally, when the e:EncryptedKey is processed, this e:EncryptedData would have been decrypted at that time and made available for processing in the event of successful Client authentication as described above. Because the contents of this payload are relevant to the rest of this example, the contents of the payload will be briefly described.

Lines (MSTS-0187)-(MSTS-0225) contain the **decrypted contents of the SOAP message s:Body element**, which contain a WS-Trust t:RequestSecurityToken element. The t:RequestType (MSTS-0189) is “…Issue”, which means this is a request to issue a security token, which is the main service of the STS. The WS-Policy wsp:AppliesTo element (MSTS-0191)-(MSTS-0198) contains the a:Address of the service, to which the Client is requesting access, a service which the STS is presumably responsible for authenticating and equipping validated clients to enable their access to. In this case the service is identified by the URL http://server.example.com/Scenarios5, which was part of the WS-SX Interop [WSSX-WSTR-WSSC-INTEROP].

Lines (MSTS-0199)-(MSTS-0205) contain the Client entropy required by the Service policy (P079), which is entropy data provided by the Client to aid in production of the ephemeral key, K2, that will be used for Client-Service communication. Lines (MSTS-0206)-(MSTS-0212) contain additional parameters used in the WS-Trust interface to the STS that will not be described here.

Lines (MSTS-0213)-(MSTS-0224) contain the WS-Trust t:SecondaryParameters, which contains the contents of the Service policy’s RequestSecurityTokenTemplate (P045)-(P059), which the Service requires that the Client pass to the STS as described above. The t:SecondaryParameters contains the information the Service has requested about the SAML 1.1 IssuedToken that the STS is being requested to create and deliver to the Client in order the enable the Client to access the Service successfully.

Assuming everything above has executed successfully, the STS will then issue the SAML 1.1 IssuedToken and return it to the Client in a WS-Trust t:RequestSecurityTokenResponse that is described next.

Here is an example STS response to the above Client request:

```
(RSTS-001)    <s:Envelope xmlns:s=http://schemas.xmlsoap.org/soap/envelope
(RSTS-002)      xmlns:a=http://www.w3.org/2005/08/addressing
(RSTS-003)      xmlns:e=http://www.w3.org/2001/04/xmlenc#
```

```
(RSTS-004)      xmlns:k="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-
              secext-1.1.xsd"
(RSTS-005)      xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
              wssecurity-secext-1.0.xsd"
(RSTS-006)      xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
              wssecurity-utility-1.0.xsd"  >
(RSTS-007)      <s:Header>
(RSTS-008)        <a:Action s:mustUnderstand="1" u:Id="_4">
(RSTS-009)       http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal
(RSTS-010)        </a:Action>
(RSTS-011)        <a:RelatesTo u:Id="_5">
(RSTS-012)          urn:uuid:04d386bf-f850-459e-918b-ad80f3d1e088
(RSTS-013)        </a:RelatesTo>
(RSTS-014)        <a:To s:mustUnderstand="1" u:Id="_6">
(RSTS-015)          http://www.w3.org/2005/08/addressing/anonymous
(RSTS-016)        </a:To>
(RSTS-017)        <o:Security s:mustUnderstand="1">
(RSTS-018)          <u:Timestamp
(RSTS-019)              u:Id="uuid-0c947d47-f527-410a-a674-753a9d7d97f7-18">
(RSTS-020)            <u:Created>2005-10-25T00:47:38.718Z</u:Created>
(RSTS-021)            <u:Expires>2005-10-25T00:52:38.718Z</u:Expires>
(RSTS-022)          </u:Timestamp>
(RSTS-023)          <e:ReferenceList>
(RSTS-024)            <e:DataReference URI="#_3"/>
(RSTS-025)          </e:ReferenceList>
(RSTS-026)          <k:SignatureConfirmation u:Id="_0"
(RSTS-027)              Value="iHGJ+xV2VZTjMlRc7AQJrwLY/aM="/>
(RSTS-028)          <k:SignatureConfirmation u:Id="_1"
(RSTS-029)              Value=
(RSTS-030)    "Ovxdeg4KQcfQlT/hEBJz+Z8dQUAfChaWIcmG3xGLZYcc8tbmCtZFuQz9tnW35
(RSTS-031)    Lmst6vRefuPA7ewRLYORAOjf92SxMbeVTlrIxQbIQNw0bs4SBSLfAo14="/>
(RSTS-032)          <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
(RSTS-033)            <SignedInfo>
(RSTS-034)              <CanonicalizationMethod Algorithm=
(RSTS-035)                  "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(RSTS-036)              <SignatureMethod Algorithm=
(RSTS-037)                  "http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
(RSTS-038)              <Reference URI="#_2">
(RSTS-039)                <Transforms>
(RSTS-040)                  <Transform Algorithm=
(RSTS-041)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(RSTS-042)                </Transforms>
(RSTS-043)                <DigestMethod Algorithm=
(RSTS-044)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(RSTS-045)                <DigestValue>kKx5bpLLlyucgXQ6exv/PbjsFlA=</DigestValue>
(RSTS-046)              </Reference>
(RSTS-047)              <Reference URI="#_4">
(RSTS-048)                <Transforms>
(RSTS-049)                  <Transform Algorithm=
(RSTS-050)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(RSTS-051)                </Transforms>
(RSTS-052)                <DigestMethod Algorithm=
(RSTS-053)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(RSTS-054)                <DigestValue>LB+VGn4fP2z45jg0Mdzyo8yTAwQ=</DigestValue>
(RSTS-055)              </Reference>
(RSTS-056)              <Reference URI="#_5">
(RSTS-057)                <Transforms>
(RSTS-058)                  <Transform Algorithm=
(RSTS-059)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(RSTS-060)                </Transforms>
(RSTS-061)                <DigestMethod Algorithm=
(RSTS-062)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(RSTS-063)                <DigestValue>izHLxm6V4Lc3PSs9Y6VRv3I5RPw=</DigestValue>
(RSTS-064)              </Reference>
```

```
(RSTS-065)              <Reference URI="#_6">
(RSTS-066)                <Transforms>
(RSTS-067)                  <Transform Algorithm=
(RSTS-068)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
                          </Transforms>
(RSTS-069)                <DigestMethod Algorithm=
(RSTS-070)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(RSTS-071)                <DigestValue>6LS4X08vC/GMGay2vwmD8fL7J2U=</DigestValue>
(RSTS-072)              </Reference>
(RSTS-073)              <Reference
(RSTS-074)                  URI="#uuid-0c947d47-f527-410a-a674-753a9d7d97f7-18">
(RSTS-075)                <Transforms>
(RSTS-076)                  <Transform Algorithm=
(RSTS-077)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(RSTS-078)                </Transforms>
(RSTS-079)                <DigestMethod Algorithm=
(RSTS-080)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(RSTS-081)                <DigestValue>uXGSpCBfbT1fLNBLdGMgy6DGDio=</DigestValue>
(RSTS-082)              </Reference>
(RSTS-083)              <Reference URI="#_0">
(RSTS-084)                <Transforms>
(RSTS-085)                  <Transform Algorithm=
(RSTS-086)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(RSTS-087)                </Transforms>
(RSTS-088)                <DigestMethod Algorithm=
(RSTS-089)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(RSTS-090)                <DigestValue>z86w+GrzqRZF56ciuz6ogzVXAUA=</DigestValue>
(RSTS-091)              </Reference>
(RSTS-092)              <Reference URI="#_1">
(RSTS-093)                <Transforms>
(RSTS-094)                  <Transform Algorithm=
(RSTS-095)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(RSTS-096)                </Transforms>
(RSTS-097)                <DigestMethod Algorithm=
(RSTS-098)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(RSTS-099)                <DigestValue>Z9TfD20a5aGngsyOPEvQuE0urvQ=</DigestValue>
(RSTS-0100)             </Reference>
(RSTS-0101)           </SignedInfo>
(RSTS-0102)          <SignatureValue>Q7HhboPUaZyXqUKgG7NCYlhMTXI=</SignatureValue>
(RSTS-0103)           <KeyInfo>
(RSTS-0104)             <o:SecurityTokenReference
(RSTS-0105)                 k:TokenType="http://docs.oasis-open.org/wss/
(RSTS-0106)                   oasis-wss-soap-message-security-1.1#EncryptedKey"
(RSTS-0107)                 xmlns:k="http://docs.oasis-open.org/wss/
(RSTS-0108)                   oasis-wss-wssecurity-secext-1.1.xsd">
(RSTS-0109)               <o:KeyIdentifier ValueType=
(RSTS-0110)   "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
              1.1.xsd#EncryptedKeySHA1">
(RSTS-0111)                 CixQW5yEb3mw6XYqD8Ysvrf8cwI=
(RSTS-0112)               </o:KeyIdentifier>
(RSTS-0113)             </o:SecurityTokenReference>
(RSTS-0114)           </KeyInfo>
(RSTS-0115)         </Signature>
(RSTS-0116)       </o:Security>
(RSTS-0117)     </s:Header>
(RSTS-0118)     <s:Body u:Id="_2">
(RSTS-0119)       <e:EncryptedData Id="_3"
(RSTS-0120)            Type="http://www.w3.org/2001/04/xmlenc#Content">
(RSTS-0121)         <e:EncryptionMethod Algorithm=
(RSTS-0122)             "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
(RSTS-0123)         <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
(RSTS-0124)           <o:SecurityTokenReference
(RSTS-0125)               k:TokenType="http://docs.oasis-open.org/wss/
(RSTS-0126)                 oasis-wss-soap-message-security-1.1#EncryptedKey"
```

```
(RSTS-0127)               xmlns:k="http://docs.oasis-open.org/wss/
(RSTS-0128)                 oasis-wss-wssecurity-secext-1.1.xsd">
(RSTS-0129)             <o:KeyIdentifier ValueType=
(RSTS-0130)   "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
              1.1.xsd#EncryptedKeySHA1">
(RSTS-0131)               CixQW5yEb3mw6XYqD8Ysvrf8cwI=
(RSTS-0132)             </o:KeyIdentifier>
(RSTS-0133)           </o:SecurityTokenReference>
(RSTS-0134)         </KeyInfo>
(RSTS-0135)         <e:CipherData>
(RSTS-0136)           <e:CipherValue>
(RSTS-0137)             <!--base64 encoded octets of encrypted RSTR-->
(RSTS-0138)             <!—
(RSTS-0139)      <t:RequestSecurityTokenResponseCollection>
(RSTS-0140)       <t:RequestSecurityTokenResponse>
(RSTS-0141)         <t:TokenType>
(RSTS-0142)   http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
              1.1#SAMLV1.1
(RSTS-0143)         </t:TokenType>
(RSTS-0144)         <t:KeySize>256</t:KeySize>
(RSTS-0145)         <t:RequestedAttachedReference>
(RSTS-0146)           <o:SecurityTokenReference>
(RSTS-0147)             <o:KeyIdentifier ValueType=
(RSTS-0148)   "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
              1.0#SAMLAssertionID">
(RSTS-0149)               uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16
(RSTS-0150)             </o:KeyIdentifier>
(RSTS-0151)           </o:SecurityTokenReference>
(RSTS-0152)         </t:RequestedAttachedReference>
(RSTS-0153)         <t:RequestedUnattachedReference>
(RSTS-0154)           <o:SecurityTokenReference>
(RSTS-0155)             <o:KeyIdentifier ValueType=
(RSTS-0156)   "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
              1.0#SAMLAssertionID">
(RSTS-0157)               uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16
(RSTS-0158)             </o:KeyIdentifier>
(RSTS-0159)           </o:SecurityTokenReference>
(RSTS-0160)         </t:RequestedUnattachedReference>
(RSTS-0161)         <t:Lifetime>
(RSTS-0162)           <u:Created>2005-10-24T20:19:26.526Z</u:Created>
(RSTS-0163)           <u:Expires>2005-10-25T06:24:26.526Z</u:Expires>
(RSTS-0164)         </t:Lifetime>
(RSTS-0165)         <t:RequestedSecurityToken>
(RSTS-0166)           <saml:Assertion MajorVersion="1" MinorVersion="1"
(RSTS-0167)             AssertionID="uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16"
(RSTS-0168)              Issuer="Test STS" IssueInstant="2005-10-24T20:24:26.526Z"
(RSTS-0169)              xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
(RSTS-0170)             <saml:Conditions NotBefore="2005-10-24T20:19:26.526Z"
(RSTS-0171)                 NotOnOrAfter="2005-10-25T06:24:26.526Z">
(RSTS-0172)               <saml:AudienceRestrictionCondition>
(RSTS-0173)                 <saml:Audience
(RSTS-0174)                  >http://server.example.com/Scenarios5</saml:Audience>
(RSTS-0175)               </saml:AudienceRestrictionCondition>
(RSTS-0176)             </saml:Conditions>
(RSTS-0177)             <saml:Advice>
(RSTS-0178)             </saml:Advice>
(RSTS-0179)             <saml:AttributeStatement>
(RSTS-0180)               <saml:Subject>
(RSTS-0181)                 <saml:SubjectConfirmation>
(RSTS-0182)                  <saml:ConfirmationMethod>
(RSTS-0183)                    urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
(RSTS-0184)                  </saml:ConfirmationMethod>
(RSTS-0185)                  <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
(RSTS-0186)                     <e:EncryptedKey
```

```
(RSTS-0187)                        xmlns:e="http://www.w3.org/2001/04/xmlenc#">
(RSTS-0188)                      <e:EncryptionMethod Algorithm=
(RSTS-0189)                    "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
(RSTS-0190)                      </e:EncryptionMethod>
(RSTS-0191)                      <KeyInfo>
(RSTS-0192)                        <o:SecurityTokenReference xmlns:o=
(RSTS-0193)   "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
              secext-1.0.xsd">
(RSTS-0194)                          <o:KeyIdentifier ValueType=
(RSTS-0195)   "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
              1.1.xsd#ThumbprintSHA1">
(RSTS-0196)                      NQM0IBvuplAtETQvk+6gn8C13wE=
(RSTS-0197)                          </o:KeyIdentifier>
(RSTS-0198)                        </o:SecurityTokenReference>
(RSTS-0199)                      </KeyInfo>
(RSTS-0200)                      <e:CipherData>
(RSTS-0201)                        <e:CipherValue>
(RSTS-0202)   EEcYjwNoYcJ+20xTYE5e/fixl5KOgzgrfaYAxkDFv/VXiuKfl084h8PmogTfM+azcgAf
(RSTS-0203)   mArVQvOyKWXRb5vmXYfVHLlhZTbXacy+nowSUNnEjp37VDbI3RJ5k6tBHF+ow0NM/P6G
(RSTS-0204)   PNZ9ZqJi11GDgWJkFsJzNZXNbbMgwuFu3cA=</e:CipherValue>
(RSTS-0205)                      </e:CipherData>
(RSTS-0206)                    </e:EncryptedKey>
(RSTS-0207)                  </KeyInfo>
(RSTS-0208)                </saml:SubjectConfirmation>
(RSTS-0209)              </saml:Subject>
(RSTS-0210)            </saml:AttributeStatement>
(RSTS-0211)            <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
(RSTS-0212)              <SignedInfo>
(RSTS-0213)                <CanonicalizationMethod Algorithm=
(RSTS-0214)                    "http://www.w3.org/2001/10/xml-exc-c14n#">
(RSTS-0215)                </CanonicalizationMethod>
(RSTS-0216)                <SignatureMethod Algorithm=
(RSTS-0217)                    "http://www.w3.org/2000/09/xmldsig#rsa-sha1">
(RSTS-0218)                </SignatureMethod>
(RSTS-0219)                <Reference
(RSTS-0220)                  URI="#uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16">
(RSTS-0221)                  <Transforms>
(RSTS-0222)                    <Transform Algorithm=
(RSTS-0223)              "http://www.w3.org/2000/09/xmldsig#enveloped-signature">
(RSTS-0224)                    </Transform>
(RSTS-0225)                    <Transform Algorithm=
(RSTS-0226)                        "http://www.w3.org/2001/10/xml-exc-c14n#">
(RSTS-0227)                    </Transform>
(RSTS-0228)                  </Transforms>
(RSTS-0229)                  <DigestMethod Algorithm=
(RSTS-0230)                      "http://www.w3.org/2000/09/xmldsig#sha1">
(RSTS-0231)                  </DigestMethod>
(RSTS-0232)                  <DigestValue
(RSTS-0233)                      >7nHBrFPsm+LEFAoV4NoQPoEl5Lk=</DigestValue>
(RSTS-0234)                </Reference>
(RSTS-0235)              </SignedInfo>
(RSTS-0236)             <SignatureValue
(RSTS-0237)   >TugV4pTIwCH87bLD4jiMgVGtkbRBt1tRlHXJArL34A/YfA4AnGBLXB4pJdUsUxMUtbQ
(RSTS-0238)   l4PoGgEsdLNg8C77peARELGP1/Tqw7T3u5zBYHxCHCiV2FWBBfeOmwJmqoaBf8XZJ4Al
(RSTS-0239)   yqPq61P61jrQjZJafpHuYpAZnZQSvsiJaBPQ=</SignatureValue>
(RSTS-0240)              <KeyInfo>
(RSTS-0241)                <o:SecurityTokenReference xmlns:o=
(RSTS-0242)   "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
              secext-1.0.xsd">
(RSTS-0243)                  <o:KeyIdentifier ValueType=
(RSTS-0244)   http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
              1.1.xsd#ThumbprintSHA1
(RSTS-0245)                    >W+rgYBmLmVEG//scD7Vo8Kq5G7I=</o:KeyIdentifier>
(RSTS-0246)                </o:SecurityTokenReference>
```

```
(RSTS-0247)               </KeyInfo>
(RSTS-0248)             </Signature>
(RSTS-0249)           </saml:Assertion>
(RSTS-0250)         </t:RequestedSecurityToken>
(RSTS-0251)         <t:RequestedProofToken>
(RSTS-0252)           <t:BinarySecret
(RSTS-0253)   u:Id="uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-14"
(RSTS-0254)   >zT8LWAUwUrIVKA/rkCr0kxlEmKAEhcB6TGWJuAgucBM=</t:BinarySecret>
(RSTS-0255)         </t:RequestedProofToken>
(RSTS-0256)       </t:RequestSecurityTokenResponse>
(RSTS-0257)      </t:RequestSecurityTokenResponseCollection>
(RSTS-0258)                     -->
(RSTS-0259)           </e:CipherValue>
(RSTS-0260)         </e:CipherData>
(RSTS-0261)       </e:EncryptedData>
(RSTS-0262)     </s:Body>
(RSTS-0263)   </s:Envelope>
```

From here on the description will be less detailed except where new concepts that have not been covered previously in this example. For instance, tracing the dsig References to their associated elements and policy requirements will not be detailed since it follows the same patterns that have just been described in the message above.

The message above is a response from the STS to the Client that contains the requested security tokens that the Client needs to access the Service.

Lines (RSTS-023)-(RSTS-025) contain **a standalone e:ReferenceList** that has an e:DataReference (Id="_3") that points to the e:EncryptedData element in the SOAP s:Body (RSTS-0119). This e:DataReference will be used later in the processing of this message.

Lines (RSTS-026)-(RSTS-031) contain **2 WS-Security 1.1 k:SignatureConfirmation elements** that indicate to the Client that the STS has processed the data in the Client request and in particular, has processed the information covered by the 2 dsig Signatures in that request, that are identified by their respective dsig SignatureValue elements (see lines (MSTS-0134)-(MSTS-0135) and (MSTS-0161)-(MSTS-0163) in the Client request to the STS above to compare SignatureValues).

Lines (RSTS-032)-(RSTS-0115) contain the **WS-SP message signature** for this message.

Lines (RSTS-0103)-(RSTS-0114) contain the **WS-SP message signature dsig KeyInfo element**, which contains a WS-Security 1.1 o:SecurityTokenReference, which contains an o:KeyIdentifier of ValueType " … EncryptedKeySHA1". This is a WS-Security 1.1 construct [WS_SECURITY_11] used to reference a security token that is not included in the message, which in this case is the Client-generated ephemeral key, K1, that the Client used to prepare the request and delivered to the STS in the e:EncryptedKey element in that request (MSTS-027) that was described in detail above. The contents of the o:KeyIdentifier (RSTS-0111) consist of the SHA1 of K1. This should be sufficient information to let the Client know that the STS used its X509 certificate, X509T2, to decrypt K1 from the e:EncryptedKey in the Client request, which will enable the Client to trust the contents of this STS response.

Lines (RSTS-0119)-(RSTS-0261) contain the **e:EncryptedData**, which is also provided in decrypted form for instructive purposes. As mentioned above, this element is referred to by the standalone e:ReferenceList element in the WS-Security header (RSTS-023) and as a result can is not tied to any particular encryption key as described in [WS_SECURITY_11], and since referenced will be processed by WS-Security.

Lines (RSTS-0123)-(RSTS-0134) contain the **e:EncryptedData dsig KeyInfo**, which refers to the same ephemeral key, K1, as described above for the message signature for this message.

Lines (RSTS-0140)-(RSTS-0256) contain the **decrypted WS-Trust t:RequestSecurityTokenResponse** from the STS. Briefly, lines (RSTS-0145)-(RSTS-0160) contain WS-Security o:SecurityTokenReference elements that refer to the SAML 1.1 Assertion that is provided below. These are convenience elements for the Client to use in subsequent message preparation where the SAML Assertion is used.

Lines (RSTS-0165)-(RSTS-0250) contain **the actual t:RequestedSecurityToken** that has been the main object of discussion up until this point, which is the SAML 1.1 Assertion, saml:Assertion (RSTS-0166)-(RSTS-0249), that is the sp:IssuedToken provided by the STS for the Client to use to access the Service.

Lines (RSTS-0170)-(RSTS-0176) of the saml:Assertion contain the **saml:Conditions** that specify that this token is intended only for the Service, identified by the URL in the **saml:Audience** element (RSTS-0174).

Lines (RSTS-0181)-(RSTS-0208) contain **the all-important saml:SubjectConfirmation element**, which contains the **STS-generated encrypted ephemeral key, K3**, that will be used by the Client and Service to communicate securely. There are **2 copies of this key, K3**, in this t:RequestSecurityTokenResponse. This copy is for the Service. The Client's copy is described below. In any event, the saml:SubjectConfirmation element contains a dsig KeyInfo (RSTS-0185)-(RSTS-0207), which contains an e:EncryptedKey element (RSTS-0186)-(RSTS-0206), which, in turn, contains a second dsig KeyInfo, which contains a WS-Security o:SecurityTokenReference element that contains an o:KeyIdentifier (RSTS-0194)-(RSTS-0197) that identifies the key, X509T3, the Service's public X509 certificate, that can be used by the Service to decrypt the ultimate object here, which is the ephemeral key, K3, contained in the e:CipherData (RSTS-0200)-(RSTS-0205). The Service's public X509 certificate is identified by its WS-Security 1.1 ThumbprintSHA1 (RSTS-0196). Therefore, when the Service receives this saml:Assertion, it has the ability to obtain the ephemeral key, K3, contained in the saml:SubjectConfirmation, with which it can securely communicate with the Client, based on the assurances provided by the STS.

Lines (RSTS-0211)-(RSTS-0248) contain the **saml:Assertion dsig Signature** that is contained in and covers the saml:Assertion via the saml:AssertionID, the value of which appears on lines (RSTS-0220) and (RSTS-0167).

Lines (RSTS-0240)-(RSTS-0247) contain the **saml:Assertion dsig Signature KeyInfo element**, which contains the ThumbprintSHA1 of the **STS public key**, **X509T2**, which is the same certificate that the Client referenced in the e:EncryptedKey when it made initial contact with the STS above (MSTS-035). This completes the initial discussion of the characteristics of the IssuedToken saml:Assertion that will be used in the remainder of this example.

Lines (RSTS-0251)-(RSTS-0255) of the t:RequestSecurityTokenResponse contains the **t:RequestedProofToken**, which contains the **Client copy of the STS-generated ephemeral key, K3**, which will be used in the Client-Service communication to authenticate the client to the Service.

At this point we have completed the setup portion of this example that has enabled secure trusted communication between the Client and Service as governed by an STS. The exact strength of the security protecting this example would be the subject of an official security analysis that is beyond the scope of this document, however, the intent has been to provide sufficient detail that all parties concerned with such an analysis would have sufficient context to understand and evaluate such an analysis.

Here is an example of a Client request to the Service using the tokens from the STS response:

```
(M001)       <s:Envelope xmlns:s=http://www.w3.org/2003/05/soap-envelope
(M002)         xmlns:a=http://www.w3.org/2005/08/addressing
(M003)         xmlns:e=http://www.w3.org/2001/04/xmlenc#
(M004)         xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
             wssecurity-secext-1.0.xsd"
(M005)         xmlns:sc="http://docs.oasis-open.org/ws-sx/ws-
             secureconversation/200512"
(M006)         xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
             wssecurity-utility-1.0.xsd"  >
(M007)         <s:Header>
(M008)           <a:Action s:mustUnderstand="1" u:Id="_5">
(M009)             http://example.org/Ping
(M010)           </a:Action>
(M011)           <a:MessageID u:Id="_6">
(M012)             urn:uuid:a859eb17-1855-4d4f-8f73-85e4cba3e423
(M013)           </a:MessageID>
(M014)           <a:ReplyTo u:Id="_7">
(M015)             <a:Address>
(M016)               http://www.w3.org/2005/08/addressing/anonymous
(M017)             </a:Address>
(M018)           </a:ReplyTo>
(M019)           <a:To s:mustUnderstand="1" u:Id="_8">
(M020)             http://server.example.com/Scenarios5
```

```
(M021)            </a:To>
(M022)            <o:Security s:mustUnderstand="1">
(M023)              <u:Timestamp
(M024)                  u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-14">
(M025)                <u:Created>2005-10-25T00:47:38.222Z</u:Created>
(M026)                <u:Expires>2005-10-25T00:52:38.222Z</u:Expires>
(M027)              </u:Timestamp>
(M028)              <e:EncryptedKey
(M029)                  Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-4">
(M030)                <e:EncryptionMethod Algorithm=
(M031)                    http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
(M032)                <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
(M033)                  <o:SecurityTokenReference>
(M034)                    <o:KeyIdentifier ValueType=
(M035)        "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
              1.1.xsd#ThumbprintSHA1">
(M036)                      NQM0IBvuplAtETQvk+6gn8C13wE=
(M037)                    </o:KeyIdentifier>
(M038)                  </o:SecurityTokenReference>
(M039)                </KeyInfo>
(M040)                <e:CipherData>
(M041)                  <e:CipherValue>
(M042)                    <!-- base64 encoded octets of encrypted key K2 -->
(M043)                  </e:CipherValue>
(M044)                </e:CipherData>
(M045)              </e:EncryptedKey>
(M046)              <sc:DerivedKeyToken u:Id="_0" >
(M047)                <o:SecurityTokenReference>
(M048)                  <o:Reference
(M049)                      URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-4"/>
(M050)                </o:SecurityTokenReference>
(M051)                <sc:Offset>0</sc:Offset>
(M052)                <sc:Length>24</sc:Length>
(M053)                <sc:Nonce>7hI6Ul6OHavffYgpquHWuQ==</sc:Nonce>
(M054)              </sc:DerivedKeyToken>
(M055)              <sc:DerivedKeyToken u:Id="_2">
(M056)                <o:SecurityTokenReference>
(M057)                  <o:Reference
(M058)                      URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-4"/>
(M059)                </o:SecurityTokenReference>
(M060)                <sc:Nonce>OEu+WEeUxPFRQK7SCFAnEQ==</sc:Nonce>
(M061)              </sc:DerivedKeyToken>
(M062)              <e:ReferenceList>
(M063)                <e:DataReference URI="#_4"/>
(M064)              </e:ReferenceList>
(M065)              <!-- encrypted SAML assertion -->
(M066)              <e:EncryptedData Id="_3"
(M067)                  Type="http://www.w3.org/2001/04/xmlenc#Element">
(M068)                <e:EncryptionMethod Algorithm=
(M069)                    "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
(M070)                <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
(M071)                  <!-- encrypted Key K2 -->
(M072)                  <e:EncryptedKey>
(M073)                    <e:EncryptionMethod Algorithm=
(M074)                        "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
(M075)                    <KeyInfo>
(M076)                      <o:SecurityTokenReference>
(M077)                        <o:KeyIdentifier ValueType=
(M078)        "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
              1.1.xsd#ThumbprintSHA1">
(M079)                          NQM0IBvuplAtETQvk+6gn8C13wE=
(M080)                        </o:KeyIdentifier>
(M081)                      </o:SecurityTokenReference>
(M082)                    </KeyInfo>
```

```
(M083)                    <e:CipherData>
(M084)                      <e:CipherValue>
(M085)
              cb7+JW2idPNSarK9quqCe9PQwmW2hoUghuyKRe+I9zOts6HaMcg73LqCWuK/jtdpvNl6
(M086)        GT/ZDYfcAJ7NLyMGxSiwi4DUlTOShqS6OTYBIKgUKiA+zXNl2koVSy7amcUhPMIT6/fo
(M087)        hH+6MZDA4t6jomcyhlCiW8d9IAzSWFkfg2k=
(M088)                      </e:CipherValue>
(M089)                    </e:CipherData>
(M090)                  </e:EncryptedKey>
(M091)                </KeyInfo>
(M092)                <e:CipherData>
(M093)                  <e:CipherValue>
(M094)                    <!-- base64 encoded octets from SAML assertion encrypted
(M095)                      with the encrypted key K2 above -->
(M096)                    <!-- SAML assertion element is identical as received in
(M097)                      RSTR , unencrypted form is omitted for brevity -->
(M098)                    <!--.........-->
(M099)                  </e:CipherValue>
(M0100)               </e:CipherData>
(M0101)             </e:EncryptedData>
(M0102)
(M0103)             <sc:DerivedKeyToken u:Id="_9">
(M0104)               <o:SecurityTokenReference>
(M0105)                 <o:KeyIdentifier ValueType=
(M0106)       "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
              1.0#SAMLAssertionID">
(M0107)                   uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16
(M0108)                 </o:KeyIdentifier>
(M0109)               </o:SecurityTokenReference>
(M0110)               <sc:Offset>0</sc:Offset>
(M0111)               <sc:Length>24</sc:Length>
(M0112)               <sc:Nonce>pgnS/VDSzJn6SFz+Vy23JA==</sc:Nonce>
(M0113)             </sc:DerivedKeyToken>
(M0114)             <Signature Id="_1" xmlns="http://www.w3.org/2000/09/xmldsig#">
(M0115)               <SignedInfo>
(M0116)                 <CanonicalizationMethod Algorithm=
(M0117)                     "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M0118)                 <SignatureMethod Algorithm=
(M0119)                     "http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
(M0120)                 <Reference URI="#_3">
(M0121)                   <Transforms>
(M0122)                     <Transform Algorithm=
(M0123)                         "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M0124)                   </Transforms>
(M0125)                   <DigestMethod Algorithm=
(M0126)                       "http://www.w3.org/2000/09/xmldsig#sha1"/>
(M0127)                   <DigestValue>eQdQVGRkVI1YfKJBw7vOYCOeLQw=</DigestValue>
(M0128)                 </Reference>
(M0129)                 <Reference URI="#_5">
(M0130)                   <Transforms>
(M0131)                     <Transform Algorithm=
(M0132)                         "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M0133)                   </Transforms>
(M0134)                   <DigestMethod Algorithm=
(M0135)                       "http://www.w3.org/2000/09/xmldsig#sha1"/>
(M0136)                   <DigestValue>xxyKpp5RZ2TebKca2IGOafIgcxk=</DigestValue>
(M0137)                 </Reference>
(M0138)                 <Reference URI="#_6">
(M0139)                   <Transforms>
(M0140)                     <Transform Algorithm=
(M0141)                         "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M0142)                   </Transforms>
(M0143)                   <DigestMethod Algorithm=
(M0144)                       "http://www.w3.org/2000/09/xmldsig#sha1"/>
```

```
(M0145)                   <DigestValue>WyGDDyYbL/hQZJfE3Yx2aK3RkK8=</DigestValue>
(M0146)                 </Reference>
(M0147)                 <Reference URI="#_7">
(M0148)                   <Transforms>
(M0149)                     <Transform Algorithm=
(M0150)                         "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M0151)                   </Transforms>
(M0152)                   <DigestMethod Algorithm=
(M0153)                       "http://www.w3.org/2000/09/xmldsig#sha1"/>
(M0154)                   <DigestValue>AEOH0t2KYR8mivgqUGDrgMtxgEQ=</DigestValue>
(M0155)                 </Reference>
(M0156)                 <Reference URI="#_8">
(M0157)                   <Transforms>
(M0158)                     <Transform Algorithm=
(M0159)                         "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M0160)                   </Transforms>
(M0161)                   <DigestMethod Algorithm=
(M0162)                       "http://www.w3.org/2000/09/xmldsig#sha1"/>
(M0163)                   <DigestValue>y8n6Dxd3DbD6TR6d6H/oVWsV4yE=</DigestValue>
(M0164)                 </Reference>
(M0165)                 <Reference
(M0166)                     URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-14">
(M0167)                   <Transforms>
(M0168)                     <Transform Algorithm=
(M0169)                         "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M0170)                   </Transforms>
(M0171)                   <DigestMethod Algorithm=
(M0172)                       "http://www.w3.org/2000/09/xmldsig#sha1"/>
(M0173)                   <DigestValue>/Cc+bGkkeQ6jlVXZx8PGgmF6MjI=</DigestValue>
(M0174)                 </Reference>
(M0175)               </SignedInfo>
(M0176)               <!--base64 encoded signature value -->
(M0177)              <SignatureValue>EyKUHUuffPUPE/ZjaFrMJJ5KLKY=</SignatureValue>
(M0178)               <KeyInfo>
(M0179)                 <o:SecurityTokenReference>
(M0180)                   <o:Reference URI="#_0"/>
(M0181)                 </o:SecurityTokenReference>
(M0182)               </KeyInfo>
(M0183)             </Signature>
(M0184)             <!-- signature over the primary signature above
(M0185)                  using the key derived from the proof-key, K3,
(M0186)                  associated with SAML assertion -->
(M0187)             <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
(M0188)               <SignedInfo>
(M0189)                 <CanonicalizationMethod Algorithm=
(M0190)                     "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M0191)                 <SignatureMethod Algorithm=
(M0192)                     "http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
(M0193)                 <Reference URI="#_1">
(M0194)                   <Transforms>
(M0195)                     <Transform Algorithm=
(M0196)                         "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M0197)                   </Transforms>
(M0198)                   <DigestMethod Algorithm=
(M0199)                       "http://www.w3.org/2000/09/xmldsig#sha1"/>
(M0200)                   <DigestValue>TMSmLlgeUn8cxyb6OYe5Q2nUuxY=</DigestValue>
(M0201)                 </Reference>
(M0202)               </SignedInfo>
(M0203)              <SignatureValue>Fh4NyOpAi+NqVFiHBgHWyvzah9I=</SignatureValue>
(M0204)               <KeyInfo>
(M0205)                 <o:SecurityTokenReference>
(M0206)                   <o:Reference URI="#_9"/>
(M0207)                 </o:SecurityTokenReference>
(M0208)               </KeyInfo>
```

```
(M0209)             </Signature>
(M0210)           </o:Security>
(M0211)         </s:Header>
(M0212)         <s:Body u:Id="_3">
(M0213)           <e:EncryptedData Id="_4"
(M0214)               Type="http://www.w3.org/2001/04/xmlenc#Content">
(M0215)             <e:EncryptionMethod Algorithm=
(M0216)                 "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
(M0217)             <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
(M0218)               <o:SecurityTokenReference >
(M0219)                 <o:Reference URI="#_2"/>
(M0220)               </o:SecurityTokenReference>
(M0221)             </KeyInfo>
(M0222)             <e:CipherData>
(M0223)               <e:CipherValue>
(M0224)                 <!-- base64 encoded octets of encrypted body content-->
(M0225)               </e:CipherValue>
(M0226)             </e:CipherData>
(M0227)           </e:EncryptedData>
(M0228)         </s:Body>
(M0229)       </s:Envelope>
```

The message above is a request from the Client to the Service using the tokens provided by the STS above.

Lines (M022)-(M0210) contain **the WS-Security o:Security header** for this request.

Lines (M028)-(M045) contain an **e:EncryptedKey that contains the ephemeral key, K2**, for the Client-Service communication. The service must use its X509T3 (M034)-(M037) to decrypt this Client-generated ephemeral key, K2.

Lines (M046)-(M054) contain a **WS-SecureConversation sc:DerivedKeyToken**, that contains the information required to **generate the signing key, DKT1(K2)** [WS_SECURE_CONV], including an sc:Nonce (M053), and a WS-Security SecurityTokenReference that contains a direct reference (M049) to the e:EncryptedKey, K2 (M042). This derived key, DKT1(K2), is used to sign the message in the message signature element (M0114)-(M0183).

Lines (M055)-(M054) provide similar **sc:DerivedKeyToken** constructs the **generate the encrypting key DKT2(K2)**. This derived key, DKT2(K2), is used to encrypt the message body, resulting in the EncryptedData element in the s:Body, on lines (M0213)-(M0227).

Lines (M062)-(M064) provide an **e:ReferenceList to reference the e:EncryptedData in the s:Body** (M0213), which contains the Client request for the Service. The s:Body payload data in this Client request is not of interest to the security properties of this example and will not be shown in decrypted form.

Lines (M066)-(M0101) contain an **e:EncryptedData element that contains the saml:Assertion** described in the STS response above. This e:EncryptedData contains a dsig KeyInfo, which, in turn, contains an e:EncryptedKey element (M072)-(M090), which contains the Client-generated ephemeral key, K2, which was used by the Client to directly encrypt the saml:Assertion. The encryption key, K2, may be decrypted, again using the Service public X509 certificate, again identified by its ThumbprintSHA1, (M077)-(M080).

Lines (M0103)-(M0113) contain a **third sc:DerivedKeyToken** that contains the information necessary for the Service to **generate the endorsing signing key, DKT3(K2)**.

Lines (M0114)-(M0183) contains **the message signature**. It is **signed using** the key identified in the dsig KeyInfo (M0178)-(M0182), which contains a reference to the **derived signing key, DKT1(K2)**, which may be constructed by the service using the sc:DerivedKeyToken (M046) described above.

Lines (M0187)-(M0209) contain the **message endorsing signature**. It is **signed using the client proof key, K3**, that was generated by the STS. Note that the Service should compare this key, K3, with the one in the saml:SubjectConfirmation in the decrypted saml:Assertion to verify that the Client is using the same proof key, K3, that is contained in the saml:Assertion that authenticates this request.

Lines (M0213)-(M0227) contain the **SOAP s:Body message payload, e:EncryptedData**, which may be decrypted using the sc:DerivedKeyToken (M055) to generate the decryption key DKT2(K2).

Here is an example response from the Service to the Client:

```
(R001)    <s:Envelope xmlns:s=http://www.w3.org/2003/05/soap-envelope
(R002)      xmlns:a=http://www.w3.org/2005/08/addressing
(R003)      xmlns:e=http://www.w3.org/2001/04/xmlenc#
(R004)      xmlns:k="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
          1.1.xsd"
(R005)      xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
          wssecurity-secext-1.0.xsd"
(R006)      xmlns:sc="http://docs.oasis-open.org/ws-sx/ws-
          secureconversation/200512"
(R007)      xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
          wssecurity-utility-1.0.xsd">
(R008)      <s:Header>
(R009)        <a:Action s:mustUnderstand="1" u:Id="_6">
(R010)          http://example.org/PingResponse
(R011)        </a:Action>
(R012)        <a:RelatesTo u:Id="_7">
(R013)          urn:uuid:a859eb17-1855-4d4f-8f73-85e4cba3e423
(R014)        </a:RelatesTo>
(R015)        <a:To s:mustUnderstand="1" u:Id="_8">
(R016)          http://www.w3.org/2005/08/addressing/anonymous
(R017)        </a:To>
(R018)        <o:Security s:mustUnderstand="1">
(R019)          <u:Timestamp u:Id="uuid-24adda3a-247a-4fec-b4f7-fb3827496cee-16">
(R020)            <u:Created>2005-10-25T00:47:38.921Z</u:Created>
(R021)            <u:Expires>2005-10-25T00:52:38.921Z</u:Expires>
(R022)          </u:Timestamp>
(R023)          <sc:DerivedKeyToken u:Id="_0" >
(R024)            <o:SecurityTokenReference
(R025)                k:TokenType="http://docs.oasis-open.org/wss/
(R026)                  oasis-wss-soap-message-security-1.1#EncryptedKey"
(R027)                xmlns:k="http://docs.oasis-open.org/wss/
(R028)                  oasis-wss-wssecurity-secext-1.1.xsd">
(R029)              <o:KeyIdentifier ValueType=
(R030)    "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
          1.1.xsd#EncryptedKeySHA1"
(R031)               >pDJlrSJLzIqi+AcQLUB4GjUuRLs=</o:KeyIdentifier>
(R032)            </o:SecurityTokenReference>
(R033)            <sc:Offset>0</sc:Offset>
(R034)            <sc:Length>24</sc:Length>
(R035)            <sc:Nonce>KFjy1Gb73BubLul0ZGgx+w==</sc:Nonce>
(R036)          </sc:DerivedKeyToken>
(R037)          <sc:DerivedKeyToken u:Id="_3">
(R038)            <o:SecurityTokenReference
(R039)                k:TokenType="http://docs.oasis-open.org/wss/
(R040)                  oasis-wss-soap-message-security-1.1#EncryptedKey"
(R041)                xmlns:k="http://docs.oasis-open.org/wss/
(R042)                  oasis-wss-wssecurity-secext-1.1.xsd">
(R043)              <o:KeyIdentifier ValueType=
(R044)    "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
          1.1.xsd#EncryptedKeySHA1"
(R045)               >pDJlrSJLzIqi+AcQLUB4GjUuRLs=</o:KeyIdentifier>
(R046)            </o:SecurityTokenReference>
(R047)            <sc:Nonce>omyh+Eg6XIa8q3V5IkHiXg==</sc:Nonce>
(R048)          </sc:DerivedKeyToken>
(R049)          <e:ReferenceList>
(R050)            <e:DataReference URI="#_5"/>
(R051)          </e:ReferenceList>
(R052)          <k:SignatureConfirmation u:Id="_1"
(R053)              Value="EyKUHUuffPUPE/ZjaFrMJJ5KLKY="/>
(R054)          <k:SignatureConfirmation u:Id="_2"
```

```
(R055)              Value="Fh4NyOpAi+NqVFiHBgHWyvzah9I="/>
(R056)          <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
(R057)            <SignedInfo>
(R058)              <CanonicalizationMethod Algorithm=
(R059)                  "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(R060)              <SignatureMethod Algorithm=
(R061)                  "http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
(R062)              <Reference URI="#_4">
(R063)                <Transforms>
(R064)                  <Transform Algorithm=
(R065)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(R066)                </Transforms>
(R067)                <DigestMethod Algorithm=
(R068)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(R069)                <DigestValue>y/oItF5TcTOFan7SavhZTTTv48M=</DigestValue>
(R070)              </Reference>
(R071)              <Reference URI="#_6">
(R072)                <Transforms>
(R073)                  <Transform Algorithm=
(R074)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(R075)                </Transforms>
(R076)                <DigestMethod Algorithm=
(R077)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(R078)                <DigestValue>X4UIaLWnaAWTriw4UJ/SFDgm090=</DigestValue>
(R079)              </Reference>
(R080)              <Reference URI="#_7">
(R081)                <Transforms>
(R082)                  <Transform Algorithm=
(R083)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(R084)                </Transforms>
(R085)                <DigestMethod Algorithm=
(R086)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(R087)                <DigestValue>vqy8/D4CDCaI1nnd4wl1Qjyp+qM=</DigestValue>
(R088)              </Reference>
(R089)              <Reference URI="#_8">
(R090)                <Transforms>
(R091)                  <Transform Algorithm=
(R092)                      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(R093)                </Transforms>
(R094)                <DigestMethod Algorithm=
(R095)                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
(R096)                <DigestValue>H11vLAr5g8pbZ6jfZ+2WNyiNjiM=</DigestValue>
(R097)              </Reference>
(R098)              <Reference
(R099)                  URI="#uuid-24adda3a-247a-4fec-b4f7-fb3827496cee-16">
(R0100)               <Transforms>
(R0101)                 <Transform Algorithm=
(R0102)                     "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(R0103)               </Transforms>
(R0104)               <DigestMethod Algorithm=
(R0105)                   "http://www.w3.org/2000/09/xmldsig#sha1"/>
(R0106)               <DigestValue>dr0g6hycoc884i+BD8FYCJGbbbE=</DigestValue>
(R0107)             </Reference>
(R0108)             <Reference URI="#_1">
(R0109)               <Transforms>
(R0110)                 <Transform Algorithm=
(R0111)                     "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(R0112)               </Transforms>
(R0113)               <DigestMethod Algorithm=
(R0114)                   "http://www.w3.org/2000/09/xmldsig#sha1"/>
(R0115)               <DigestValue>Rv3N7VNfAqpn0khr3F/qQZmE/l4=</DigestValue>
(R0116)             </Reference>
(R0117)             <Reference URI="#_2">
(R0118)               <Transforms>
```

```
(R0119)                <Transform Algorithm=
(R0120)                    "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(R0121)              </Transforms>
(R0122)              <DigestMethod Algorithm=
(R0123)                  "http://www.w3.org/2000/09/xmldsig#sha1"/>
(R0124)              <DigestValue>X2pxEnYPM8cMLrbhNqPgs8xk+a4=</DigestValue>
(R0125)           </Reference>
(R0126)          </SignedInfo>
(R0127)          <SignatureValue>I2jQuDTWWQiNJy/ziyg8AFYO/z4=</SignatureValue>
(R0128)          <KeyInfo>
(R0129)            <o:SecurityTokenReference>
(R0130)              <o:Reference URI="#_0"/>
(R0131)            </o:SecurityTokenReference>
(R0132)          </KeyInfo>
(R0133)        </Signature>
(R0134)      </o:Security>
(R0135)    </s:Header>
(R0136)    <s:Body u:Id="_4">
(R0137)      <e:EncryptedData Id="_5"
(R0138)          Type="http://www.w3.org/2001/04/xmlenc#Content">
(R0139)        <e:EncryptionMethod Algorithm=
(R0140)            "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
(R0141)        <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
(R0142)          <o:SecurityTokenReference>
(R0143)            <o:Reference URI="#_3"/>
(R0144)          </o:SecurityTokenReference>
(R0145)        </KeyInfo>
(R0146)        <e:CipherData>
(R0147)          <e:CipherValue>
(R0148)            <!--base64 encoded octets of encrypted body content-->
(R0149)          </e:CipherValue>
(R0150)        </e:CipherData>
(R0151)      </e:EncryptedData>
(R0152)    </s:Body>
(R0153)  </s:Envelope>
```

The message above is a response from the Service to the Client using the tokens based on the saml:Assertion IssuedToken from the STS.

Lines (R018)-(R0134) contain the WS-Security o:Security header in the SOAP s:Header.

Lines (R023)-(R036) contain an **sc:DerivedKeyToken** that may be used to **construct the derived signing key DKT4(K2)**, which uses the Client-generated ephemeral key, K2, that the Service received in the Client request (M028)-(M045) above, and is now used in the response to the client, similar to the way the STS used K1 to respond to the Client, except that in this case the Service will use derived keys DKT4(K2) and DKT5(K2) in addition to K2 in the response. This sc:DerivedKeyToken contains a WS-Security o:SecurityTokenReference (R024)-(R032) that uses the WS-Security 1.1 mechanism EncryptedKeySHA1 to identify the Client-generated ephemeral key, K2, as the key to use to derive DKT4(K2) along with the sc:Nonce provided (R035) (and the label as described in [WS_SECURE_CONV]).

Lines (R037)-(R048) contain a **second sc:DerivedKeyToken** that may be used by the Client **to construct the derived encryption key, DKT5(K2)**. The same EncryptedKeySHA1 mechanism is used by the Client to construct DKT5(K2) as described for DKT4(K2).

Lines (R049)-(R051) contain an **e:ReferenceList** containing an e:DataReference to the EncryptedData element in the s:Body (R0136).

Lines (R052)-(R055) contain 2 WS-Security 1.1 k:SignatureConfirmation elements confirming the dsig Signatures that were in the client request above (M0203) on the endorsing signature and (M0177) on the message signature of the Client request. The dsig SignatureValues compare respectively to assure the Client that the data from the Client request was processed and is what is being referred to in this response.

Lines (R056)-(R0134) contain the message signature, which is signed as referenced in the dsig KeyInfo (R0128)-(R0132) by DKT4(K2), which may be constructed as described above using the sc:DerivedKeyToken element (R023)-(R036).

Lines (R0137)-(R0151) contain the Service response payload to the Client which may be decrypted using DKT5(K2) using the sc:DerivedKeyToken element (R037)-(R048).

2.4 Secure Conversation Scenarios

2.4.1 (WSS 1.0) Secure Conversation bootstrapped by Mutual Authentication with X.509 Certificates

This scenario corresponds to the situation where both parties have an X.509 certificate (and public/private key pair). Because of the volume of messages from each source, the Requestor/Initiator uses WS-SecureConversation to establish a new session key and uses the session key for integrity and/or confidentiality protection. This improves performance, by using less expensive symmetric key operations and improves security by reducing the exposure of the long term secret.

The recipient models this scenario by using the symmetric binding that includes a SecureConversationToken assertion to describe the token type accepted by this endpoint. This token assertion further contains a bootstrap policy to indicate the security binding that is used by requestors that want a security context token issued by this service. The bootstrap policy affects the Request Security Token Request (RST) and Request Security Token Request Response (RSTR) messages sent between the Initiator and the Recipient to establish the security context. It is modeled by use of an asymmetric binding assertion for this scenario because both parties mutually authenticate each other using their X.509 certificates.

The message level policies cover a different scope of the web service definition than the security binding level policy and so appear as separate policies and are attached at Message Policy Subject. These are shown below as input and output policies. Thus, we need a set of coordinated policies one with endpoint subject (WSS10SecureConversation_policy) and two (WSS10SecureConversation_input_policy, WSS10SecureConversation_output_policy) with message subjects to achieve this use case.

```
(P001) <wsp:Policy wsu:Id="WSS10SecureConversation_policy">
(P002)   <wsp:ExactlyOne>
(P003)     <wsp:All>
(P004)       <sp:SymmetricBinding xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
(P005)         <wsp:Policy>
(P006)           <sp:ProtectionToken>
(P007)             <wsp:Policy>
(P008)               <sp:SecureConversationToken
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken
/AlwaysToRecipient">
(P009)                 <wsp:Policy>
(P010)                   <sp:RequireDerivedKeys/>
(P011)                   <sp:BootstrapPolicy>
(P012)                     <wsp:Policy>
(P013)                       <wsp:ExactlyOne>
(P014)                         <wsp:All>
(P015)                           <sp:AsymmetricBinding
             xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
(P016)                             <wsp:Policy>
(P017)                               <sp:InitiatorToken>
(P018)                                 <wsp:Policy>
(P019)                                   <sp:X509Token
sp:IncludeToken="http://schemas.xmlsoap.org/ws/200512/securitypolicy/IncludeToken/
AlwaysToRecipient">
(P020)                                     <wsp:Policy>
(P021)                                       <sp:WssX509V3Token10/>
(P022)                                     </wsp:Policy>
```

```
(P023)                                    </sp:X509Token>
(P024)                                  </wsp:Policy>
(P025)                                </sp:InitiatorToken>
(P026)                                <sp:RecipientToken>
(P027)                                  <wsp:Policy>
(P028)                                    <sp:X509Token
sp:IncludeToken="http://schemas.xmlsoap.org/ws/200512/securitypolicy/IncludeToken/
AlwaysToInitiator">
(P029)                                      <wsp:Policy>
(P030)                                        <sp:WssX509V3Token10/>
(P031)                                      </wsp:Policy>
(P032)                                    </sp:X509Token>
(P033)                                  </wsp:Policy>
(P034)                                </sp:RecipientToken>
(P035)                                <sp:AlgorithmSuite>
(P036)                                  <wsp:Policy>
(P037)                                    <sp:TripleDesRsa15/>
(P038)                                  </wsp:Policy>
(P039)                                </sp:AlgorithmSuite>
(P040)                                <sp:Layout>
(P041)                                  <wsp:Policy>
(P042)                                    <sp:Strict/>
(P043)                                  </wsp:Policy>
(P044)                                </sp:Layout>
(P045)                                <sp:IncludeTimestamp/>
(P046)                                <sp:OnlySignEntireHeadersAndBody/>
(P047)                              </wsp:Policy>
(P048)                            </sp:AsymmetricBinding>
(P049)                            <sp:Wss10>
(P050)                              <wsp:Policy>
(P051)                                <sp:MustSupportRefKeyIdentifier/>
(P052)                              </wsp:Policy>
(P053)                            </sp:Wss10>
(P054)                            <sp:SignedParts>
(P055)                              <sp:Body/>
(P056)                              <sp:Header Name="Action"
                            Namespace="http://www.w3.org/2005/08/addressing"/>
(P057)                            </sp:SignedParts>
(P058)                            <sp:EncryptedParts>
(P059)                              <sp:Body/>
(P060)                            </sp:EncryptedParts>
(P061)                          </wsp:All>
(P062)                        </wsp:ExactlyOne>
(P063)                      </wsp:Policy>
(P064)                    </sp:BootstrapPolicy>
(P065)                  </wsp:Policy>
(P066)                </sp:SecureConversationToken>
(P067)              </wsp:Policy>
(P068)            </sp:ProtectionToken>
(P069)            <sp:AlgorithmSuite>
(P070)              <wsp:Policy>
(P071)                <sp:Basic256/>
(P072)              </wsp:Policy>
(P073)            </sp:AlgorithmSuite>
(P074)            <sp:Layout>
(P075)              <wsp:Policy>
(P076)                <sp:Strict/>
(P077)              </wsp:Policy>
(P078)            </sp:Layout>
(P079)            <sp:IncludeTimestamp/>
(P080)            <sp:OnlySignEntireHeadersAndBody/>
(P081)          </wsp:Policy>
(P082)        </sp:SymmetricBinding>
(P083)        <sp:Trust13>
```

```
(P084)          <wsp:Policy>
(P085)            <sp:RequireClientEntropy/>
(P086)            <sp:RequireServerEntropy/>
(P087)          </wsp:Policy>
(P088)        </sp:Trust13>
(P089)
(P090)      </wsp:All>
(P091)    </wsp:ExactlyOne>
(P092) </wsp:Policy>

(P093) <wsp:Policy wsu:Id="WSS10SecureConversation_input_policy">
(P094)   <wsp:ExactlyOne>
(P095)     <wsp:All>
(P096)       <sp:SignedParts>
(P097)         <sp:Header Name="Action"
                            Namespace="http://www.w3.org/2005/08/addressing"/>
(P098)         <sp:Header Name="To"
                            Namespace="http://www.w3.org/2005/08/addressing"/>
(P099)         <sp:Header Name="MessageID"
                            Namespace="http://www.w3.org/2005/08/addressing"/>
(P0100)              <sp:Body/>
(P0101)            </sp:SignedParts>
(P0102)          </wsp:All>
(P0103)   </wsp:ExactlyOne>
(P0104)  </wsp:Policy>
(P0105)
(P0106) <wsp:Policy wsu:Id="WSS10SecureConversation_output_policy">
(P0107)   <wsp:ExactlyOne>
(P0108)     <wsp:All>
(P0109)       <sp:SignedParts>
(P0110)         <sp:Header Name="Action"
                           Namespace="http://www.w3.org/2005/08/addressing"/>
(P0111)         <sp:Header Name="To"
                           Namespace="http://www.w3.org/2005/08/addressing"/>
(P0112)         <sp:Header Name="MessageID"
                           Namespace="http://www.w3.org/2005/08/addressing"/>
(P0113)         <sp:Header Name="RelatesTo"
                           Namespace="http://www.w3.org/2005/08/addressing"/>
(P0114)         <sp:Body/>
(P0115)       </sp:SignedParts>
(P0116)     </wsp:All>
(P0117)   </wsp:ExactlyOne>
(P0118) </wsp:Policy>
```

Lines (P004) – (P082) indicate that the service uses the symmetric security binding to protect messages, using a Security Context Token (Lines (P008) – (P066)) to sign messages in both directions. The actual basis for the signatures should be keys derived from that Security Context Token as stated by the RequireDerivedKey assertion in Line (P010). Messages must include a Timestamp element (Line (P079)) and the signature value must be calculated over the entire body and header elements (Line (P080)).

Lines (P083) – (P088) contain the Trust13 assertion which defines the requirements for the WS-Trust related message exchange as part of the SC bootstrap. Line (P085) indicates that the Initiator (Client) has to provide entropy to be used as key material for the requested proof token. Line (P086) requires the same from the recipient (Server) which results in computed key from both client and server entropy values.

Lines (P011) – (P065) contain the bootstrap policy for the initial creation of the security context between the communication parties before it is being used. It contains an AsymmetricBinding assertion (Lines (P015) – (P048)) which indicates that the initiator's (WS-Security 1.0 X.509 Certificate) token (Lines (P017) – (P025)) used by the recipient to verify the signature in the RST must be included in the message (P019). The exchange of entropy and key material in the body of the RST and RSTR messages is protected by the EncryptedParts assertion of the bootstrap policy in lines (P058) – (P061).

According to the MustSupportKeyRefIdentitier assertions in Line (P051), an X.509 Key Identifier must be used to identify the token. Lines (P054) – (P057) require that the body and the WS-Addressing Action header is signed on each message (RST, RSTR) within the bootstrap process.

An example of an RST message sent from the initiator to the recipient according to the bootstrap policy defined by this policy is as follows:

```
(M001) <?xml version="1.0" encoding="utf-8" ?>
(M002) <soap:Envelope xmlns:soap="..." xmlns:wsse="..." xmlns:wsu="..."
                             xmlns:wst="..." xmlns:xenc="..." xmlns:wsa="...">
(M003)   <soap:Header>
(M004)     <wsa:Action wsu:Id="action">
(M005)           http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT
(M006)         </wsa:Action>
(M007)     <wsse:Security>
(M008)       <wsu:Timestamp wsu:Id="timestamp">
(M009)         <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
(M010)         <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
(M011)       </wsu:Timestamp>
(M012)       <wsse:BinarySecurityToken wsu:Id="clientToken"
                      ValueType="...#X509v3" EncodingType="...#Base64Binary">
(M013)                 MIICZDCCAc2gAwIBAgIRALSOLzt7...
(M014)       </wsse:BinarySecurityToken>
(M015)       <ds:Signature xmlns:ds="...">
(M016)         <ds:SignedInfo>
(M017)           <ds:CanonicalizationMethod
                         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M018)           <ds:SignatureMethod
                      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
(M019)           <ds:Reference URI="#action">
(M020)             <ds:Transforms>
(M021)               <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-
                                                                exc-c14n#"/>
(M022)             </ds:Transforms>
(M023)             <ds:DigestMethod
                          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(M024)             <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
(M025)           </ds:Reference>
(M026)           <ds:Reference URI="#timestamp">...</ds:Reference>
(M027)           <ds:Reference URI="#body">...</ds:Reference>
(M028)         </ds:SignedInfo>
(M029)         <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
(M030)         <ds:KeyInfo>
(M031)           <wsse:SecurityTokenReference>
(M032)             <wsse:Reference URI="#clientToken"
                                                ValueType="...#X509v3"/>
(M033)           </wsse:SecurityTokenReference>
(M034)         </ds:KeyInfo>
(M035)       </ds:Signature>
(M036)       <xenc:EncryptedKey>
(M037)         <xenc:EncryptionMethod Algorithm="...#rsa-1_5"/>
(M038)         <ds:KeyInfo>
(M039)           <wsse:SecurityTokenReference>
(M040)             <wsse:KeyIdentifier
(M041)               ValueType="...#X509v3SubjectKeyIdentifier">AtETQ...
(M042)             </wsse:KeyIdentifier>
(M043)           </wsse:SecurityTokenReference>
(M044)         </ds:KeyInfo>
(M045)         <xenc:CipherData>
(M046)           <xenc:CipherValue>
(M047)             <!-- encrypted key -->
(M048)           </xenc:CipherValue>
(M049)         </xenc:CipherData>
(M050)         <xenc:ReferenceList>
(M051)           <xenc:DataReference URI="#request"/>
```

```
(M052)         </xenc:ReferenceList>
(M053)       </xenc:EncryptedKey>
(M054)     </wsse:Security>
(M055)   </soap:Header>
(M056)   <soap:Body wsu:Id="body">
(M057)     <xenc:EncryptedData Id="request" Type="...#Content">
(M058)       <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
(M059)         <xenc:CipherData>
(M060)           <xenc:CipherValue>
(M061)             <!-- encrypted RST request -->
(M062)     <!-- ### Begin unencrypted RST request
(M063)     <wst:RequestSecurityToken>
(M064)       <wst:TokenType>
(M065)          http://docs.oasis-open.org/ws-sx/ws-secureconversation/
                                                                 200512/sct
(M066)       </wst:TokenType>
(M067)       <wst:RequestType>
(M068)          http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
(M069)       </wst:RequestType>
(M070)       <wsp:AppliesTo
                    xmlns:wsp="http://www.w3.org/ns/ws-policy">
(M071)         <wsp:EndpointReference>
(M072)           <wsp:Address>
(M073)              http://acme.com/ping/
(M074)           </wsp:Address>
(M075)         </wsp:EndpointReference>
(M076)       </wsp:AppliesTo>
(M077)       <wst:Entropy>
(M078)         <wst:BinarySecret>cr9bOcb1wCEyY9ehaGK33e41h9s=
(M079)         </wst:BinarySecret>
(M080)       </wst:Entropy>
(M081)     </wst:RequestSecurityToken>
(M082)     ### End unencrypted RST request -->
(M083)         </xenc:CipherValue>
(M084)       </xenc:CipherData>
(M085)     </xenc:EncryptedData>
(M086)   </soap:Body>
(M087) </soap:Envelope>
```

Line (M005) indicates to the recipient that the SCT Binding of WS-Trust is used.

Lines (M008) – (M011) hold the Timestamp element as required by the IncludeTimestamp assertion.

Lines (M012) – (M014) contain the initiator's X.509 token as required by the InitiatorToken assertion's value ("…/AlwaysToRecipient").

Lines (M015) – (M035) hold the message signature.

Lines (M036) – (M053) hold the encrypted symmetric key used to encrypt the RST request in the body of the message according to the bootstrap policy. It contains a Subject Key Identifier of the X.509 Certificate used to encrypt the key and not the certificate itself as required by the RecipientToken assertion's value ("…/Never") in (P028).

Lines (M019) – (M025) indicate the WS-Addressing Action header is included in the signature as required by the SignedParts assertion.

Line (M026) indicates the Timestamp is included in the signature according to the IncludeTimestamp assertion definition.

Line (M027) indicates the SOAP Body is included in the signature as required by the SignedParts assertion.

Lines (M056) – (M086) hold the Security Token Reference pointing to the initiators X.509 certificate.

Lines (M038) – (M058) contain the SOAP Body of the message with the encrypted RST element. Lines (M062) – (M082) show the unencrypted content of the RST request. It specifies the Token Type (Lines (M064) – (M066)) and the Request Type (Lines (M067) – (M069). According to the Trust13 assertion, it also includes entropy provided by the initiator as indicated by Lines (M077) – (M080).

An example of an RSTR message sent from the recipient to the initiator according to the bootstrap policy defined by this policy is as follows:

```
(M001)    <?xml version="1.0" encoding="UTF-8"?>
(M002)    <soap:Envelope xmlns:soap="..." xmlns:wst="..." xmlns:wsc="..."
                   xmlns:xenc="...">
(M003)      <soap:Header>
(M004)        <wsa:Action wsu:Id="action">
(M005)          http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTRC/IssueFinal
(M006)        </wsa:Action>
(M007)        <wsse:Security>
(M008)          <wsu:Timestamp wsu:Id="timestamp">
(M009)            <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
(M010)            <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
(M011)          </wsu:Timestamp>
(M012)          <wsse:BinarySecurityToken wsu:Id="serviceToken"
                      ValueType="...0#X509v3" EncodingType="...#Base64Binary">
(M013)                   MIIASDPIOASDsaöAgIRALSOLzt7...
(M014)                </wsse:BinarySecurityToken>
(M015)          <ds:Signature xmlns:ds="...">
(M016)            <ds:SignedInfo>
(M017)              <ds:CanonicalizationMethod
                         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M018)              <ds:SignatureMethod
                      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
(M019)              <ds:Reference URI="#action">
(M020)                <ds:Transforms>
(M021)                  <ds:Transform
                         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M022)                </ds:Transforms>
(M023)                <ds:DigestMethod
                          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(M024)                <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
(M025)              </ds:Reference>
(M026)              <ds:Reference URI="#timestamp">...</ds:Reference>
(M027)              <ds:Reference URI="#body">...</ds:Reference>
(M028)            </ds:SignedInfo>
(M029)            <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
(M030)            <ds:KeyInfo>
(M031)              <wsse:SecurityTokenReference>
(M032)                <wsse:Reference URI="#myToken" ValueType="...#X509v3"/>
(M033)              </wsse:SecurityTokenReference>
(M034)            </ds:KeyInfo>
(M035)          </ds:Signature>
(M036)          <xenc:EncryptedKey>
(M037)            <xenc:EncryptionMethod Algorithm="...#rsa-1_5"/>
(M038)              <ds:KeyInfo>
(M039)                <wsse:SecurityTokenReference>
(M040)                  <wsse:KeyIdentifier
(M041)                      ValueType="...#X509v3SubjectKeyIdentifier">AtETQ...
(M042)                  </wsse:KeyIdentifier>
(M043)                </wsse:SecurityTokenReference>
(M044)              </ds:KeyInfo>
(M045)            <xenc:CipherData>
(M046)              <xenc:CipherValue>
(M047)                 <!-- encrypted key -->
(M048)              </xenc:CipherValue>
(M049)            </xenc:CipherData>
(M050)            <xenc:ReferenceList>
(M051)              <xenc:DataReference URI="#response"/>
(M052)             </xenc:ReferenceList>
(M053)           </xenc:EncryptedKey>
(M054)        </wsse:Security>
(M055)      </soap:Header>
```

```
(M056)      <soap:Body wsu:Id="body">
(M057)        <xenc:EncryptedData Id="response" Type="...#Content">
(M058)          <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
(M059)            <xenc:CipherData>
(M060)              <xenc:CipherValue>
(M061)                <!-- encrypted RSTR -->
(M062)        <!-- ### Begin unencrypted RSTR
(M063)        <wst:RequestSecurityTokenResponseCollection>
(M064)          <wst:RequestSecurityTokenResponse>
(M065)            <wst:RequestedSecurityToken>
(M066)              <wsc:SecurityContextToken>
(M067)                <wsc:Identifier>uuid:...</wsc:Identifier>
(M068)              </wsc:SecurityContextToken>
(M069)            </wst:RequestedSecurityToken>
(M070)            <wst:RequestedProofToken>
(M071)              <xenc:EncryptedKey Id="ek049ea390c90011dbba4e00304852867e">
(M072)                <xenc:EncryptionMethod 
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
(M073)                <ds:KeyInfo>
(M074)                  <wsse:SecurityTokenReference>
(M075)                    <wsse:KeyIdentifier 
                             ValueType="...#X509SubjectKeyIdentifier" 
                             EncodingType="...#Base64Binary">
(M076)                       Wjw2gDCBye6NJAh0lPCyUldvTN8=
(M077)                    </wsse:KeyIdentifier>
(M078)                  </wsse:SecurityTokenReference>
(M079)                </ds:KeyInfo>
(M080)                <xenc:CipherData>
(M081)                  <xenc:CipherValue>hhx9TcaVL6XwBNl...</xenc:CipherValue>
(M082)                </xenc:CipherData>
(M083)              </xenc:EncryptedKey>
(M084)            </wst:RequestedProofToken>
(M085)            <wsp:AppliesTo 
                    xmlns:wsp="http://www.w3.org/ns/ws-policy">
(M086)              <wsp:EndpointReference>
(M087)                <wsp:Address>
(M088)                   http://acme.com/ping/
(M089)                </wsp:Address>
(M090)              </wsp:EndpointReference>
(M091)            </wsp:AppliesTo>
(M092)            <wst:Entropy>
(M093)              <wst:BinarySecret>asdhjwkjqwe123498SDFALasd=
(M094)              </wst:BinarySecret>
(M095)            </wst:Entropy>
(M096)          </wst:RequestSecurityTokenResponse>
(M097)        </wst:RequestSecurityTokenResponseCollection>
(M098)        ### End unencrypted RSTR -->
(M099)            </xenc:CipherValue>
(M0100)         </xenc:CipherData>
(M0101)       </xenc:EncryptedData>
(M0102)     </soap:Body>
(M0103)   </soap:Envelope>
```

Line (M005) indicates to the initiator that th is is the final response to the RST in an RSTRC.

Lines (M008) – (M011) hold the Timestamp element as required by the IncludeTimestamp assertion.

Lines (M012) – (M014) contain the recipient's X.509 token as required by the RecipientToken assertion's value ("…/AlwaysToInitiator").

Lines (M015) – (M035) hold the message signature.

Lines (M036) – (M053) hold the encrypted symmetric key used to encrypt the RSTR response in the body of the message according to the bootstrap policy.

Lines (M019) – (M025) indicate the WS-Addressing Action header is included in the signature as required by the SignedParts assertion.

Line (M026) indicates the Timestamp is included in the signature according to the IncludeTimestamp assertion definition.

Line (M027) indicates the SOAP Body is included in the signature as required by the SignedParts assertion.

Lines (M031) – (M033) hold the Security Token Reference pointing to the requestor’s X.509 certificate.

Lines (M056) – (M102) contain the SOAP Body of the message with the encrypted RSTR collection element. Commented out is the unencrypted form of the RSTR collection in Lines (M063) – (M097), which includes one RSTR (Lines (M064) – (M096)). Lines (M065) – (M069) contain the new Security Context Token. The accompanying RequestedProofToken in Lines (M070) – (M084) include the encrypted secret key (Lines (M071) – (M083)) of the security context. The key is encrypted using the initiator’s public key that was already used to verify its signature in the incoming request.

According to the Trust13 assertion, the response includes entropy provided by the recipient as indicated by Lines (M092) – (M095).

An Example of an SCT-secured application message sent from the initiator to the recipient according to the message input policy (WSS10SecureConversation_input_policy) is as follows:

```
(M001)    <?xml version="1.0" encoding="UTF-8"?>
(M002)    <soap:Envelope xmlns:soap="..." xmlns:wsse="..." xmlns:wsu="..."
             xmlns:wst="..." xmlns:xenc="..." xmlns:wsa="..." xmlns:wsrm="...">
(M003)      <soap:Header>
(M004)        <wsa:To wsu:Id="to">http://acme.com/ping/</wsa:To>
(M005)        <wsa:MessageID wsu:Id="msgid">
(M006)               http://acme.com/guid/1231873ledh-CA47-1067-B31D-10662DA
(M007)        </wsa:MessageID>
(M008)        <wsa:Action wsu:Id="action">urn:Ping</wsa:Action>
(M009)        <wsse:Security>
(M010)          <wsu:Timestamp wsu:Id="timestamp">
(M011)            <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
(M012)            <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
(M013)          </wsu:Timestamp>
(M014)          <wsc:SecurityContextToken wsu:Id="SCT">
(M015)            <wsc:Identifier>uuid:91f50600-60cc-11da-8e67-000000000000
(M016)                   </wsc:Identifier>
(M017)          </wsc:SecurityContextToken>
(M018)          <wsc:DerivedKeyToken wsu:Id="DKT">
(M019)            <wsse:SecurityTokenReference>
(M020)              <wsse:Reference URI="#SCT" ValueType="http://docs.oasis-
                             open.org/ws-sx/ws-secureconversation/200512/sct"/>
(M021)            </wsse:SecurityTokenReference>
(M022)            <wsc:Offset>0</wsc:Offset>
(M023)            <wsc:Length>24</wsc:Length>
(M024)            <wsc:Label>
                     WSSecure ConversationWSSecure Conversation
(M025)            </wsc:Label>
(M026)            <wsc:Nonce>ylN04kFBJesy2U2SQL6ezI3SCak=</wsc:Nonce>
(M027)          </wsc:DerivedKeyToken>
(M028)          <ds:Signature xmlns:ds="...">
(M029)            <ds:SignedInfo>
(M030)              <ds:CanonicalizationMethod
                          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(M031)              <ds:SignatureMethod
                       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
(M032)              <ds:Reference URI="#msgid">
(M033)                <ds:Transforms>
(M034)                  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-
                                                                    exc-c14n#"/>
(M035)                </ds:Transforms>
(M036)                <ds:DigestMethod
                           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(M037)                <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
(M038)              </ds:Reference>
```

```
(M039)              <ds:Reference URI="#to">...</ds:Reference>
(M040)              <ds:Reference URI="#action">...</ds:Reference>
(M041)              <ds:Reference URI="#timestamp">...</ds:Reference>
(M042)              <ds:Reference URI="#body">...</ds:Reference>
(M043)            </ds:SignedInfo>
(M044)            <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
(M045)            <ds:KeyInfo>
(M046)              <wsse:SecurityTokenReference>
(M047)                <wsse:Reference URI="#DKT" ValueType="http://docs.oasis-
                              open.org/ws-sx/ws-secureconversation/200512/dk"/>
(M048)              </wsse:SecurityTokenReference>
(M049)            </ds:KeyInfo>
(M050)          </ds:Signature>
(M051)        </wsse:Security>
(M052)      </soap:Header>
(M053)      <soap:Body wsu:Id="body">
(M054)        <pns:Ping xmlns:pns="http://tempuri.org/">
(M055)          <pns:Text>abc</pns:Text>
(M056)        </pns:Ping>
(M057)      </soap:Body>
(M058)    </soap:Envelope>
```

Lines (M004) – (M008) contain the WS-Addressing headers according to the UsingAddressing assertion.

Lines (M010) – (M013) hold the Timestamp as specified by the IncludeTimestamp assertion within the SymmetricBinding assertion.

Lines (M014) – (M017) contain the Security Context Token which has been issued by the recipient in the previous message. Based on this token, the initiator included a Derived Key Token (Lines (M018) – (M027)) as indicated by the RequireDerivedKey assertion in the policy.

Lines (M028) – (M050) hold the message signature that uses the Derived Key Token (Lines (M046) – (M048)) to sign the WS-Addressing headers (Lines (M032) – (M040)), the timestamp (Line (M041)) and the body (Line (M042)) of the message, according to the SignedParts assertion of the message input policy (WSS10SecureConversation_input_policy).

3 Conformance

This document contains non-normative examples of usage of WS-SecurityPolicy and other related specifications.

Therefore **there are no conformance statements that apply to this document**.

Refer to the referenced specifications [References], which will individually contain conformance requirements for WS-SecurityPolicy and related specifications.

A. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Ashok Malhotra, Oracle
Prateek Mishra, Oracle
Ramana Turlapati, Oracle
Martin Gudgin, Microsoft
Marc Goodner, Microsoft
Paul Cotton, Microsoft
Hal Lockhart, BEA
Symon Chang, BEA
Frederick Hirsch, Nokia
Greg Whitehead, Hewlett-Packard
Ching-Yun (C.Y.) Chao, IBM
Anthony Nadalin, IBM
Tony Gullotta, SOA

B. Revision History

[optional; should not be included in OASIS Standards]

Revision	Date	Editor	Changes Made
0.09	23-Feb-07	Rich Levinson	Updated doc format to latest OASIS template, added Symon Chang's encrypted UsernameToken scenario
0.10	6-Mar-07	Rich Levinson Tony Gullotta Symon Chang Martin Raepple	Added sample messages and explanatory text to several examples. Line numbered each example w Pxxx for the Policy, Mxxx for the sample message. Intent is to do all examples, this version is to get feedback along with progress as each example stands alone. Completed examples: 2.1.1.2, 2.1.2.1, 2.1.3, 2.1.4, 2.2.1, and 2.5.1. Partially completed examples that have sample messages: 2.1.1.1, 2.1.1.3, 2.3.1.2, 2.3.1.4, 2.3.1.5, and 2.3.2.2, 2.3.2.4, 2.3.2.5