Application Note: Using WS-Trust for Simple and Protected Negotiation Protocol

September 7, 2007

Authors

Jan Alexander, Microsoft

Vijay Gajjala, Microsoft

Kirill Gavrylyuk, Microsoft

Chris Kaler, Microsoft

Michael McIntosh, IBM

Anthony Nadalin, IBM

Bruce Rich, IBM

T.R.Vishwanath, Microsoft

Copyright Notice

Abstract

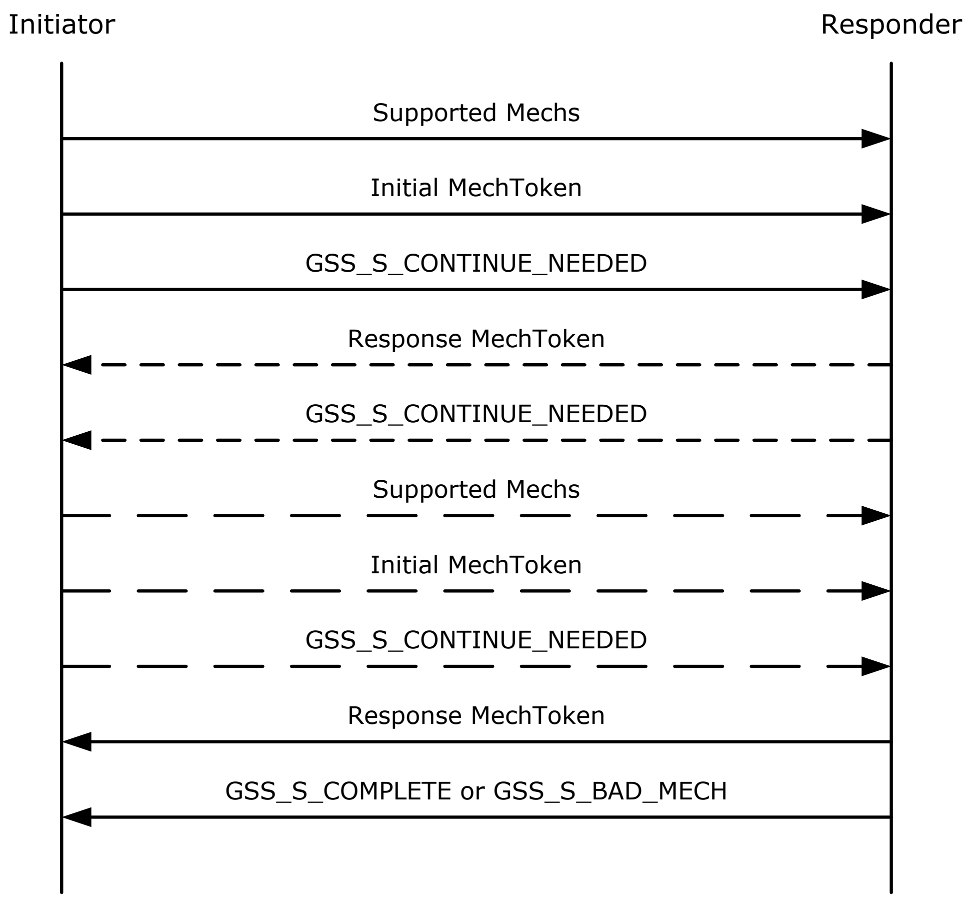
This note describes usage of WS-Trust binary negotiation framework for Simple Protected Negotiation Protocol defined in RFC 2478 [2478] to securely establish a common security mechanism as well as a shared security context between two GSS peers. For the implementers convenience, the usage is described in a form of a profile of the [WS-Trust] and [WS-SecureConversation] specifications.

Table of Contents

1. Overview

The simple and protected negotiation protocol (defined in RFC 2478) is a pseudo-security mechanism, traditionally used when two applications based on GSS-API implementations wish to establish a common security mechanism. This is most useful when the GSS peers support multiple security mechanisms.

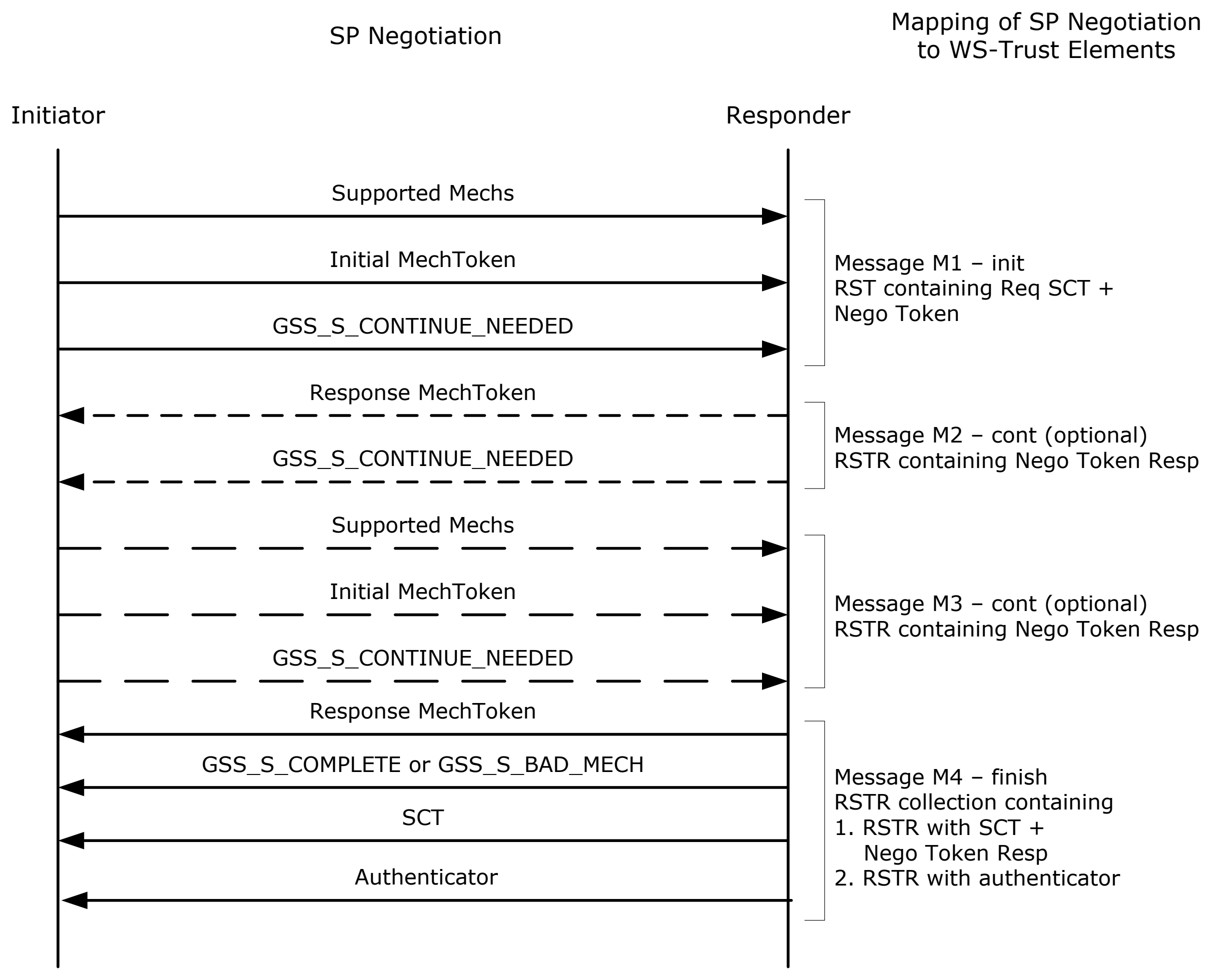
Once a common security mechanism is identified, the security mechanism might also negotiate mechanism specific options during its context establishment. Such a process is invisible to SP negotiation. SP negotiation promotes an optimistic model in which the initiator’s security token (mechToken) for the preferred security mechanism is embedded in the initial step (negotiation token) of the negotiation. This sequence is represented below:



The WS-Trust specification defines a request response pattern for security token acquisition. In addition, it describes extensions to enable exchanges for negotiation and challenges. Specifically, the BinaryExchange mechanism described in WS-Trust can be used to contain existing binary exchange formats in WS-Trust framework.

When using WS-Trust framework for SP Negotiation Protocol, WS-Trust BinaryExchange mechanism is used to carry SP Negotiation tokens of the SP Negotiation protocol to establish a common security mechanism as well as a shared security context between the two GSS peers. This document describes in detail how the SP negotiation tokens and mechanism tokens can be carried in the elements defined by WS-Trust. The end result of the negotiation protocol is the mechanism specific establishment of a shared security context between the initiator and responder.

The following diagram gives an overview of the SP Negotiation procedure in terms of WS-Trust message exchanges:



Motivations for using WS-Trust with SP-Nego protocol

1. Protocol re-use: SP Negotiation protocol has established a sequence of message exchanges and defined the contents of those message exchanges. It provides authentication facilities based on reuse of GSS API based exchanges. The same negotiation exchanges can be used for Web Services.
2. Support SOAP processing models: For example allow intermediaries during negotiation.
3. Cryptographic security: SP Negotiation protocol uses GSS API for exchanging mechanism specific security tokens and for establishing a shared security context. Use of WS-Trust and WS-Security provides necessary security mechanisms to protect negotiation of the common mechanism and establishment of a shared security context between communicating WS security peers following SP Negotiation protocol. WS-SecureConversation allows using established shared security context for Web Services security mechanisms defined in WS-Security.

2. Notations and Terminology

This section specifies the notations, namespaces, and terminology used in this document.

2.1 Notational Conventions

The keywords “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.

Namespace URIs of the general form “some-URI” represents some application-dependent or context-dependent URI as defined in RFC3986.

2.2 Namespace

The following namespaces are used in this document:

Prefix	**Namespace**
s11	http://schemas.xmlsoap.org/soap/envelope/
s12	http://www.w3.org/2003/05/soap-envelope/
s	Either s11 or s12 namespace
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wst12	http://schemas.xmlsoap.org/ws/2005/02/trust
wst13	http://docs.oasis-open.org/ws-sx/ws-trust/200512
wst	Either wst12 or wst13 namespace
wsc12	http://schemas.xmlsoap.org/ws/2005/02/sc
wsc13	http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512
wsc	Either wsc12 or wsc13 namespace
xenc	http://www.w3.org/2001/04/xmlenc#
wsa	http://www.w3.org/2005/08/addressing

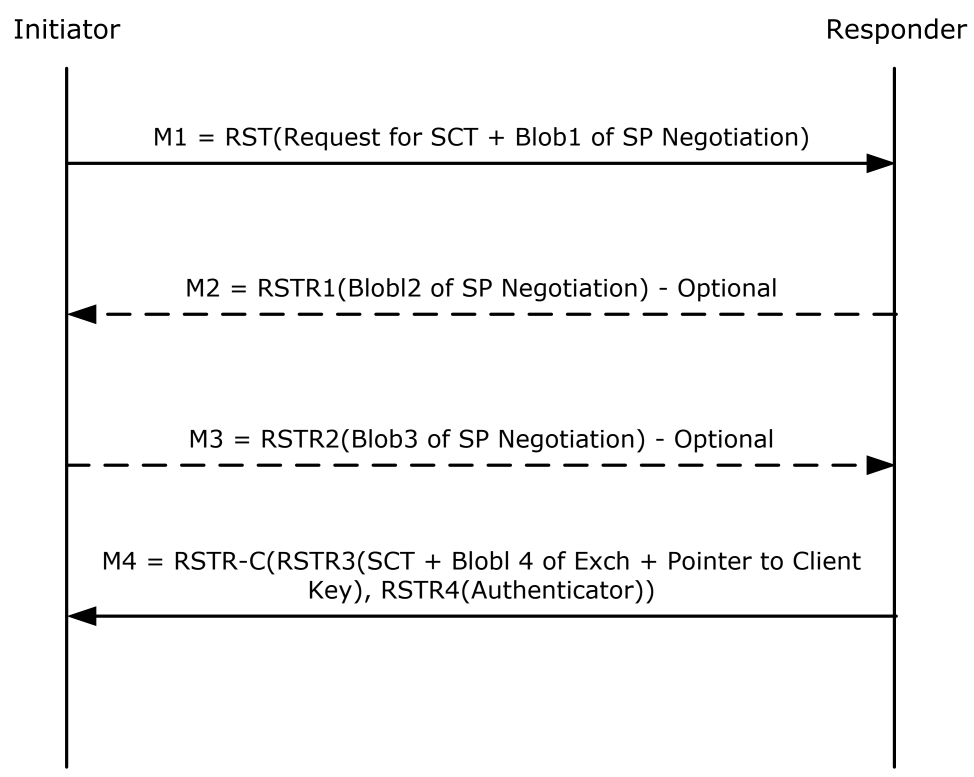
3. Negotiation

This document describes the following steps following SP-Nego protocol:

1. Establishment of a common security mechanism between the initiator and the responder.
2. Exchange of mechanism specific authentication tokens.
3. Establishment of a shared security context token. Establishment of a shared security context consisting of a shared secret.

The negotiation involves the following steps:

1. Send an RST containing a binary exchange and a request for an Security Context Token using a value type introduced in this document. The binary exchange is formulated as a base 64 encoding of the binary negotiation token obtained by making a GSS-API GSS_Init_sec_context call.
2. The GSS responder performs a GSS_Accept_sec_context with the negotiation token from the first request. If the mechanism is not accepted, or a mechanism specific exchange is required, the responder returns with a RSTR containing the base 64 encoded binary negotiation token.
3. The GSS initiator performs a GSS_Init_sec_context with the negotiation token from the previous response. The resultant response is sent as a base 64 encoded RSTR. Steps 2 and 3 are repeated until the GSS_Accept_sec_context() results in a GSS_S_COMPLETE or GSS_S_BAD.
4. The GSS responder performs a GSS_Accept_sec_context() and returns with a RSTR collection. The first RSTR contains a security context token representing the shared security context between the initiator and the responder and the second RST containing an authenticator. The proof token contains an issued key. The issued key is secured using the SSPI context.


Initiator
Responder
M1 = RST(Request for SCT + Blob1 of SP Negotiation)
M2 = RSTR1(Blobl2 of SP Negotiation) - Optional
M3 = RSTR2(Blob3 of SP Negotiation) - Optional
M4 = RSTR-C(RSTR3(SCT + Blobl 4 of Exch + Pointer to Client Key), RSTR4(Authenticator))

The reminder of this section focuses on the message details for the steps defined above. For the implementer's convenience, the usage is described in a form of a profile of the WS-Trust and WS-SecureConversation specifications.

Message 1: Initiating the negotiation

The first leg of the negotiation places the following constraints on the use of the WS-Trust RequestSecurityToken element.

1.1 RequestSecurityToken Context attribute

R1101 *This URI attribute is used to specify the correlation context for different legs of the negotiation. It MUST be provided on the first leg of negotiation and copied to all subsequent legs related to the negotiation. Note that this context provides no ordering semantics.*

1.2 RequestSecurityToken TokenType

R1201 *The TokenType element MUST be present in the RequestSecurityToken element.*

R1202 *The TokenType element MUST contain a value of*

```
http://schemas.xmlsoap.org/ws/2005/02/sc/sct
```

or the following value if the WS-SecureConversation 1.3 is used

```
http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct
```

1.3 RequestSecurityToken BinaryExchange EncodingType

R1301 The required EncodingType attribute on BinaryExchange element in RequestSecurityToken element MUST contain a value of

```
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary
```

1.4 RequestSecurityToken BinaryExchange ValueType

R1401 The required ValueType attribute on BinaryExchange element in RequestSecurityToken element MUST contain a value of

```
http://schemas.xmlsoap.org/ws/2005/02/trust/spnego
```

The value of the BinaryExchange is the base64 encoded negotiation token / handle resulting from a GSS_Inti_sec_context() call at the initiator of the negotiation.

1.5 RequestSecurityToken RequestType

R1501 The required RequestType element in RequestSecurityToken MUST contain

```
http://schemas.xmlsoap.org/ws/2005/02/trust/Issue
```

or the following value if the WS-Trust 1.3 is used

```
http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
```

1.6 RequestSecurityToken KeySize

R1601 RequestSecurityToken MUST contain KeySize element specifying the desired key size for the key that will be issued on the final leg of the negotiation.

1.7 RequestSecurityToken AppliesTo

R1701 RequestSecurityToken MAY contain AppliesTo element in accordance with WS-Trust

1.8 WS-Addressing Action URI

R1801 If the SOAP message contains wsa:Action header, the wsa:Action element MUST contain

```
http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue
```

or the following value if the WS-Trust 1.3 is used

```
http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue
```

The following is an example of the first leg of negotiation.

```
<s:Envelope>
      <s:Header>
            ...
      </s:Header>
      <s:Body>
            <wst:RequestSecurityToken Context="uuid:a4799798...">
                  <wst:TokenType>.../sct</wst:TokenType>
                  <wst:RequestType> .../Issue </wst:RequestType>
                  <wst:KeySize>256</wst:KeySize>
                  <wst:BinaryExchange
                        EncodingType=".../#Base64Binary"
                        ValueType=".../spnego">
                        FgMBAEEBAAA9A...
                  </wst:BinaryExchange>
            </wst:RequestSecurityToken>
      </s:Body>
</s:Envelope>
```

Message 2: Continued negotiation - responder

The continued negotiation – responder leg of the negotiation is optional. If the negotiation is complete or failed (i.e., the GSS_Accept_sec_context() call returns a GSS_S_COMPLETE or GSS_S_BAD_MECH) this step is not required. The following step places the following constraints on the use of RSTR.

2.1 RequestSecurityTokenResponse Context attribute

R2101 *This URI attribute MUST be copied from the initial leg (RST) of the negotiation. Note that this context provides no ordering semantics.*

2.2 RequestSecurityToken BinaryExchange

R2201 *The BinaryExchange element must be present in the RequestSecurityTokenResponse element. It must contain Base64 encoded blob of the second leg of SP Negotiation.*

2.3 RequestSecurityToken BinaryExchange EncodingType

R2301 *The required EncodingType attribute on BinaryExchange element in RequestSecurityTokenResponse element MUST contain a value of*

```
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary
```

2.4 RequestSecurityToken BinaryExchange ValueType

R2401 The *required ValueType attribute on BinaryExchange element in RequestSecurityTokenResponse element MUST contain a value of*

```
http://schemas.xmlsoap.org/ws/2005/02/trust/spnego
```

The value of the BinaryExchange is the base64 encoded negotiation token / handle resulting from a GSS_Accept_sec_context() call at the responder of the negotiation.

2.5 WS-Addressing Action URI

R2501 *If the SOAP message contains wsa:Action header, the wsa:Action element MUST contain*

```
http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Issue
```

or the following value if the WS-Trust 1.3 is used

```
http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/Issue
```

The following is an example of the second leg of negotiation.

```
<s:Envelope>
      <s:Header>
            ...
      </s:Header>
      <s:Body>
            <wst:RequestSecurityTokenResponse
                  Context=" uuid:a4799798…" >
                  <wst:BinaryExchange
                        EncodingType="...#Base64Binary"
                        ValueType=".../spnego">
                              FgMBDf0CAAB...
                  </wst:BinaryExchange>
            </wst:RequestSecurityTokenResponse>
      </s:Body>
</s:Envelope>
```

Message 3: Continued negotiation - initiator

The continued negotiation – initiator leg of the negotiation is optional. The following step places the following constraints on the use of RSTR.

3.1 RequestSecurityTokenResponse Context attribute

R3101 *This URI attribute MUST be copied from the initial leg (RST) of the negotiation. Note that this context provides no ordering semantics.*

3.2 RequestSecurityToken BinaryExchange

R3201 *The BinaryExchange element must be present in the RequestSecurityTokenResponse element. The value of this element MUST be a Base64 encoded blob of the third leg of SP Negotiation.*

3.3 RequestSecurityToken BinaryExchange EncodingType

R3301 *The required EncodingType attribute on BinaryExchange element in RequestSecurityTokenResponse element MUST contain a value of*

```
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary
```

3.4 RequestSecurityToken BinaryExchange ValueType

R3401 *The required ValueType attribute on BinaryExchange element in RequestSecurityTokenResponse element MUST contain a value of*

```
http://schemas.xmlsoap.org/ws/2005/02/trust/spnego
```

The value of the BinaryExchange is the base64 encoded negotiation token / handle resulting from a GSS_Init_sec_context() call at the initiator of the negotiation.

3.5 WS-Addressing Action URI

R3501 *If the SOAP message contains wsa:Action header, the wsa:Action element MUST contain*

```
http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Issue
```

or the following value if the WS-Trust 1.3 is used

```
http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/Issue
```

The following is an example of the third leg of negotiation

```


<s:Envelope xmlns:s=".../soap-envelope">
      <s:Header>
              ...
      </s:Header>
      <s:Body xmlns:s=".../soap-envelope">
            <wst:RequestSecurityTokenResponse
                  Context="uuid:a4799798...">
                  <wst:BinaryExchange
                        EncodingType="...#Base64Binary"
                        ValueType="spnego">
                              FgMBDf0CAAB...
```

```
                </wst:BinaryExchange>
            </wst:RequestSecurityTokenResponse>
      </s:Body>
</s:Envelope>
```

Message 4: Negotiation complete

If the negotiation is complete or failed (i.e., the GSS_Accept_sec_context() call returns a GSS_S_COMPLETE or GSS_S_BAD_MECH) this step is performed. The following section details the constraints on the Section 10.9 Authenticating Exchanges from WS-Trust

4.1 RequestSecurityTokenResponseCollection element

R4101 *The RequestSecurityTokenResponseCollection element MUST be used to contain one RSTR with the Security context token being returned and the second RSTR with the authenticator.*

R4102 *The ordering of RSTRs MUST be as specified: first RSTR MUST contain the Security context token being returned and second RSTR MUST contain the authenticator for the exchange.*

R4103 The @Context attribute used on *both RSTRs in the collection MUST match the context element from the initiation leg of the negotiation.*

4.2 Returning security context token

R4201 *The required TokenType element in the first RequestedSecurityToken element MUST contain*

```
  http://schemas.xmlsoap.org/ws/2005/02/sc/sct
```

or the following value if the WS-SecureConversation 1.3 is used

```
http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct
```

R4202 *Both RequestedAttachedReference and RequestedUnattachedReference elements MUST be present in the first RequestedSecurityToken element.*

4.3 Returning the proof key associated with the security context token

R4301 *The RequestedProofToken element specifies the proof-of-possession token associated with the requested security token and MUST be present in the response. The responder to the negotiation generates a new key and returns the key as an EncryptedKey in the proof token.*

R4302 *The xenc: EncryptedKey element MUST be present in the RequestedProofToken.*

R4303 *The EncryptionMethod element MUST be present in the RequestedProofToken element and MUST contain an Algorithm identifier*

```
http://schemas.xmlsoap.org/2005/02/trust/spnego#GSS_Wrap
```

R4304 *Issued Key*

The responder is responsible for issuing the key associated with the TLSNego session. If the initiator requested properties for the generated key (e.g. key size) in the initial RST message, the generated key SHOULD match those requirements. The issued key MUST be communicated back to the initiator using the wst:RequestedProofToken element and MUST be protected with the negotiated "master secret" using the negotiated data wrapping algorithm.

This key is contained in the <CipherData><CipherValue>...</CipherValue></CipherData> elements of the EncryptedKey.

4.4 Specifying the lifetime of the issued token

R4401 *The LifeTime element MUST be included*

R4402 *The LifeTime element MUST contain the creation time and the expiration time of the security context token. These values MUST be specified in the Created and Expires elements of LifeTime element.*

R4403 *The Created and Expires values MUST be specified in UTC format as specified by XML schema date Time type and MUST NOT generate leap seconds.*

4.5 Specifying the issued key length

R4501 *The issued key length MUST be included in the response using the wst:KeySize element.*

4.6 Specifying the exchange mode for the finish leg

R4601 *The BinaryExchange element MAY be present in the RequestSecurityTokenResponse element if it is necessary to transmit the SP Negotiation blob on the final leg. It MUST contain Base64 encoded blob of the final leg of SP Negotiation.*

4.7 Specifying the EncodingType for the binary exchange of the finish leg

R4701 *The required EncodingType attribute on BinaryExchange element in RequestSecurityTokenResponse element MUST contain a value of*

```
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary
```

4.8 RequestSecurityToken BinaryExchange ValueType

R4801 *The required ValueType attribute on BinaryExchange element in RequestSecurityTokenResponse element MUST contain a value of*

```
http://schemas.xmlsoap.org/ws/2005/02/trust/spnego
```

4.9 Proving to the requestor that the issuer knows the key, by means of an Authenticator

R4901 *The authenticator element of WS-Trust MUST be present and provides a mechanism for the issuer to prove to the requestor that it knows the key (and that the returned metadata is valid) prior to the requestor using the data.*

R4902 *The CombinedHash element MUST be present and provides the actual authenticator value. This is done by providing the base64 encoding of the first 256 bits of the PSHA1 digest of the issued key and the concatenation of SHA1 hash of the message exchanges and the string "AUTH-HASH". Specifically, PSHA1(*issued-key, *H + "AUTH-HASH")*$_{0\text{-}255}$*, where H = SHA1(ExcC14N(RST...RSTR-C))*

4.10 WS-Addressing Action URI

R41001 *If the SOAP message contains wsa:Action header, the wsa:Action element MUST contain*

```
http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Issue
```

or the following value if the WS-Trust 1.3 is used

```
http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal
```

The following is an example of the negotiation complete step using WS-Trust 1.2 and WS-SecureConversation 1.2.

```
<s:Envelope xmlns:s=".../soap-envelope">
      <s:Header> <ReplyTo>... </ReplyTo> </s:Header>
      <s:Body xmlns:s=".../soap-envelope">
            <wst:RequestSecurityTokenResponseCollection
                  xmlns:wst=".../trust">
                  <wst:RequestSecurityTokenResponse
                        Context="uuid:a4799798..." >
                        <wst:TokenType>
               http://schemas.xmlsoap.org/ws/2004/01/security/sc/sct
                        </wst:TokenType>
                        <wst:RequestedSecurityToken>
                              <wsc:SecurityContextToken
                                    wsu:Id="uuid:a4796849...">
                                    <wsu:Identifier>uuid:…</wsu:Identifier>
                              </wsc:SecurityContextToken>
                        </wst:RequestedSecurityToken>
                        <wst:RequestedAttachedReference>
                          <wsse:SecurityTokenReference>
                                  <wsse:Reference
                 ValueType=http://schemas.xmlsoap.org/ws/2005/02/sc/sct
                 URI="#uuid-901eb2cc-bbed-48ff-9deb-e6b14846e3ed-1"/>
                         </wsse:SecurityTokenReference>
                        </wst:RequestedAttachedReference>
                        <wst:RequestedUnattachedReference>
                          <wsse:SecurityTokenReference>
                           <wsse:Reference
                  URI="urn:uuid:41bd2d6a-b5ea-4025-8d33-7512fbf9f3f4"
                  ValueType=http://schemas.xmlsoap.org/ws/2005/02/sc/sct/>
                          </wsse:SecurityTokenReference>
                        </wst:RequestedUnattachedReference>
                        <wst:RequestedProofToken>
                       <xenc:EncryptedKey>
                                  <xenc:EncryptionMethod
      Algorithm="http://schemas.xmlsoap.org/2005/02/trust/spnego#GSS_Wrap" />
                                      <xenc:CipherData>
                                           <xenc:CipherValue>
FwMBACgGCoXa7cHbQ0a2drmWd4wmqCYGfiCSbNjt6slR2ZFWI8r9CH+i1gdE
                                           </xenc:CipherValue>
                                      </xenc:CipherData>
                             </xenc:EncryptedKey>
                        </wst:RequestedProofToken>
                        <wst:Lifetime>
                              <wsu:Created>…</wsu:Created>
                              <wsu:Expires>…</wsu:Expires>
                        </wst:Lifetime>
                        <wst:KeySize>256</wst:KeySize>
                  </wst:RequestSecurityTokenResponse>
                  <wst:RequestSecurityTokenResponse
                       Context="uuid:a4799798..." >
                        <wst:Authenticator>
                              <wst:CombinedHash>
```

```
                                                   ...
                                             </wst:CombinedHash>
                                       </wst:Authenticator>
                                 </wst:RequestSecurityTokenResponse>
                           </wst:RequestSecurityTokenResponseCollection>
      </s:Body>
</s:Envelope>
```

5. Security Considerations

It is critical that all relevant elements of a message be included in signatures. As well, the signatures for security context establishment must include a timestamp, nonce, or sequence number depending on the degree of replay prevention required. Security context establishment should include full policies to prevent possible attacks (e.g. downgrading attacks).

Authenticating services are susceptible to denial of service attacks. Care should be taken to mitigate such attacks as is warranted by the service.

There are many other security concerns that one may need to consider in security protocols. The list above should not be used as a "check list" instead of a comprehensive security analysis.

In addition to the consideration identified here, readers should also review the security considerations in [WS-Security], [WS-Trust12] and [WS-Trust13].

6. References

[Kerb]
J. Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5)," RFC1510, September 1993.

[KEYWORDS]
S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997

[SOAP]
W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.

[SOAP12]
W3C Recommendation, "SOAP 1.2 Part 1: Messaging Framework", 24 June 2003.

[URI]
T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 3986, MIT/LCS, Day Software, Adobe Systems, January 2005.

[WS-Security]
OASIS Standard, "Web Services Security: SOAP Message Security," 15 March 2004.

[WS-Trust12]
S.Anderson, et.al., "Web Services Trust Language (WS-Trust)," February 2005.

[WS-Trust13]

OASIS Standard, "Web Services Trust Language," 19 March 2007

[WS-SecureConversation12]
S.Anderson, et.al.,"Web Services Secure Conversation Language (WS-SecureConversation)," February 2005

[WS-SecureConversation13]
OASIS Standard, "Web Services Secure Conversation Language (WS-SecureConversation)," 1, March 2007

[2478]
E. Baize, D. Pinkas, The Simple and Protected GSS-API Negotiation Mechanism, RFC 2478, December 1998.

[1964]
J. Linn , The Kerberos Version 5 GSS-API Mechanism, RFC 1964, June 1996.

[4121]
L, Zhu, K. Jaganathan, S. Hartman, The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2, RFC 4121, July 2005.

[WS-Addressing]
W3C Recommendation, "Web Service Addressing (WS-Addressing)", 9 May 2006.