Application Note: Using WS-Trust for TLS Handshake

September 7, 2007

Authors

Jan Alexander, Microsoft

Giovanni Della-Libera, 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 the TLS Handshake protocol defined in RFC2246 to securely establish recipient's identity, securely establish a shared security context between two SOAP nodes, and to optionally establish authenticity of the sender using sender's WS-Security credentials. Such a shared security context can be used for future message privacy and integrity between those two SOAP nodes. The binary exchange mechanisms defined in WS-Trust are used to carry TLS exchange information.

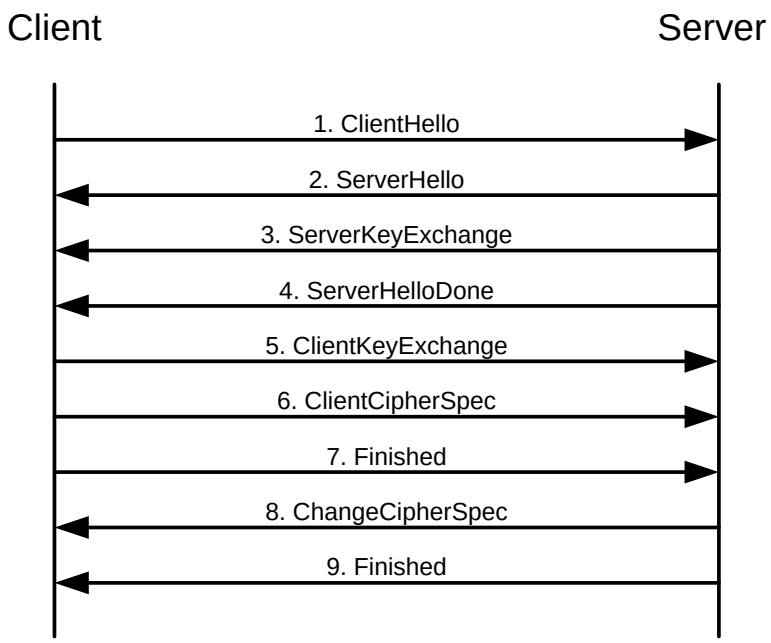
Status

Draft

Table of Contents

1. Overview

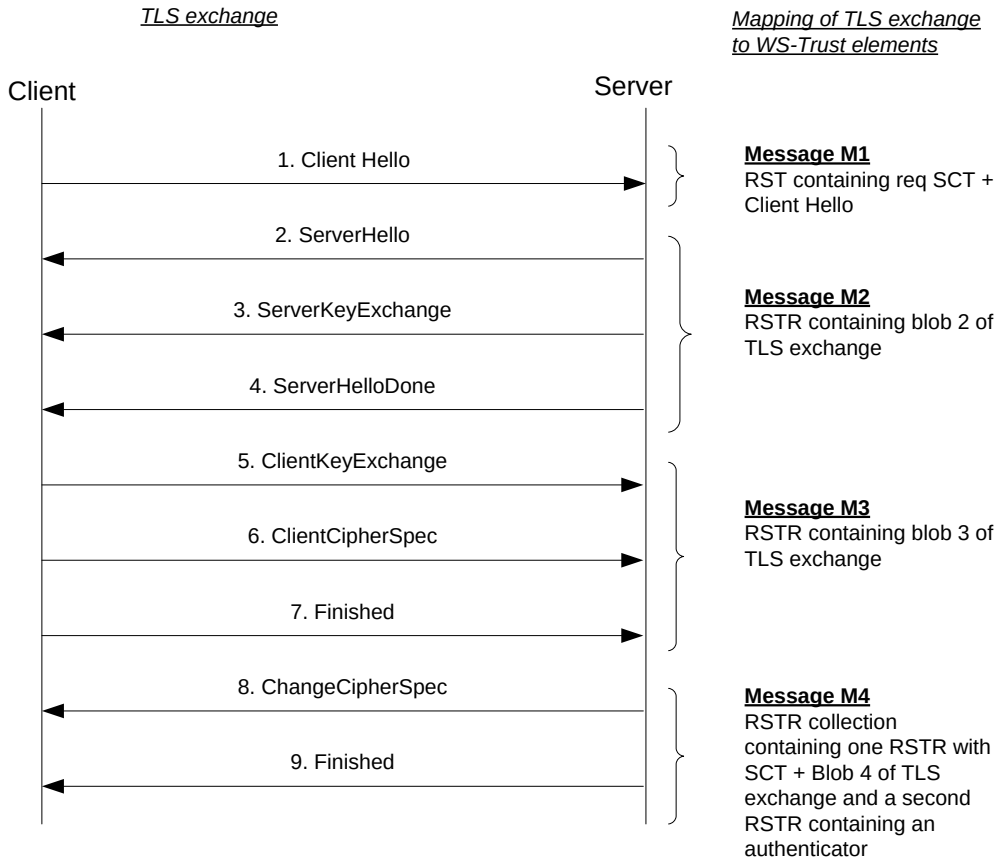
The TLS handshake protocol defines a sequence of message exchanges that establishes peer's identity and establishes a secured communication channel.



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. As such, the BinaryExchange mechanism described in WS-Trust can be used to contain existing binary exchange formats in WS-Trust framework.

When using WS-Trust for TLS Handshake Protocol the binary messages defined by TLS exchange are base 64 encoded and carried in the elements defined by WS-Trust.

The following diagram gives an overview of the TLS Handshake carried by the WS-Trust messages:



Motivations for using WS-Trust for TLS Handshake protocol

1. Cryptographic security: establish a security context between two SOAP entities with keying material which is used in securing subsequent messages between these two end points. Provide mechanisms for recipient authentication, requestor authentication.

2. Protocol re-use: TLS handshake protocol has established a sequence of message exchanges and defined the contents of those message exchanges. It provides authentication facilities. We MUST be able to use the same message exchanges.

3. Support SOAP processing models: Specifically, allow for intermediaries during negotiation.

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

The TLS handshake consists of two parts:

1. Server authentication and security context token (SCT, [WS-SecureConversation12] and [WS-SecureConversation13]) establishment
2. Client authentication and amending claims in the SCT established in part 1 above.

The server authentication phase, illustrated in the diagram above, encapsulates the TLS exchanges as base 64 encoded BinaryExchange elements of an RST/RSTR sequence.

The reminder of this section focuses on the message details for the server authentication phase. For the convenience of implementer, the usage is described in a form of a profile of the WS-Trust and WS-SecureConversation specifications. Each message exchange involves Client and Server roles used in the diagrams above.

Message 1: Initiating the negotiation

The first leg of the negotiation places the following constraints on the use of RST.

1.1 RequestSecurityToken Context attribute

This URI attribute is used to specify the correlation context for different legs of the negotiation.This context provides no ordering semantics.

R1101 *Client MUST include the Context attribute on the first Request Security Token message*

1.2 RequestSecurityToken TokenType

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

R1202 *The TokenType 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
```

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/tlsnego
```

1.5 RequestSecurityToken RequestType

R1501 *The required RequestType element 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 TLS 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=".../tlsnego">
                        FgMBAEEBAAA9A...
                  </wst:BinaryExchange>
            </wst:RequestSecurityToken>
```

```
      </s:Body>
</s:Envelope>
```

Message 2: Continued negotiation - requestor

The second leg of the negotiation 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 for TLSNego. It must contain Base64 encoded blob of the second leg of TLS exchange.*

2.3 RequestSecurityToken BinaryExchange EncodingType

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

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

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 TLS Negotiation.

```
<s:Envelope>
      <s:Header>
            ...
      </s:Header>
```

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

Message 3: Continued negotiation - initiator

The third leg of the negotiation places the following constraints on the use of RSTR.

3.1 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.*

3.2 RequestSecurityToken BinaryExchange

R2201 *The BinaryExchange element must be present in the RequestSecurityTokenResponse element for TLSNego. It must contain Base64 encoded blob of the third leg of TLS exchange.*

3.3 RequestSecurityToken BinaryExchange EncodingType

R2301 *The required EncodingType attribute on BinaryExchange element in RequestSecurityTokenResponse element for TLSNego 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

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

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

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 TLS 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=".../tlsnego">
                              FgMBDf0CAAB...
                  </wst:BinaryExchange>
            </wst:RequestSecurityTokenResponse>
      </s:Body>
</s:Envelope>
```

Message 4: Negotiation complete

The following section details the constraints on the final leg of negotiation, 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 above: first RSTR to contain the Security context token being returned and second RSTR contains the authenticator for the exchange. Specified ordering of RSTRs MUST be applied.*

R4103 The @Context attribute *MUST be copied from the initial leg (RST) of the negotiation. Note that this context provides no ordering semantics.*

4.2 Returning security context token

R4201 *The TokenType element MUST be present in the first RequestSecurityTokenResponse element of the RequestSecurityTokenResponseCollection.*

R4202 *The TokenType 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
```

R4203 Both RequestedAttachedReference and RequestedUnattachedReference elements MUST be present *in the first RequestSecurityTokenResponse element of the RequestSecurityTokenResponseCollection.*

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.*

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

R4303 *The EncryptionMethod element MUST be present in the xenc:EncryptedKey element.*

R4304 The EncryptionMethod MUST contain *an Algorithm identifier with the value of*

```
http://schemas.xmlsoap.org/2005/02/trust/tlsnego#TLS_Wrap
```

R4305 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 TLS "master secret" using the TLS data wrapping algorithm.

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

4.4 Specifying the issued key size

R4401 *The issued key length MUST be included in the response using the wst:KeySize element.*

4.5 Specifying the lifetime of the issued token

R4501 *The LifeTime element MUST be included in the RequestSecurityTokenResponse element*

R4502 *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.*

R4503 *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.*

4.6 Specifying the exchange mode for the finish leg

R4601 *The BinaryExchange element MUST be present in the RequestSecurityTokenResponse element for TLSNego. It must contain Base64 encoded blob of the final leg of TLS exchange.*

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

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

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

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(*computed-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 fourth leg of TLS Negotiation 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/2005/02/sc/sct
                        </wst:TokenType>
                        <wst:RequestedSecurityToken>
                              <wsc:SecurityContextToken wsu:Id="...">
                                    <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/tlsnego#TLS_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:BinaryExchange
                              EncodingType="...#Base64Binary"
                              ValueType=".../tlsnego">
                                    FgMBDf0CAAB...
                        </wst:BinaryExchange>
                  </wst:RequestSecurityTokenResponse>
                  <wst:RequestSecurityTokenResponse
                       Context="uuid:a4799798..." >
                        <wst:Authenticator>
                              <wst:CombinedHash>
                                    ...
                              </wst:CombinedHash>
                        </wst:Authenticator>
                  </wst:RequestSecurityTokenResponse>
            </wst:RequestSecurityTokenResponseCollection>
      </s:Body>
</s:Envelope>
```

7. 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].

8. References

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

[TLS]
T. Dierks, C. Allen, "The Transport Layer Security (TLS) Protocol, Version 1.0," RFC 2246, January 1999

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