Webサービスセキュリティの最新動向

― WS-SecurityとWS-SX（WS-SecureExchange）関連仕様（2）

佐藤史子 （日本 IBM　東京基礎研究所）

Web サービスセキュリティは，SOAP メッセージそのものを署名・暗号化することにより安全なメッセージ交換を実現するものです．本稿では，Web サービスセキュリティの基本仕様である WS-Security（OASIS Web Services Security Specification）[1] とその関連仕様（WS-SecureExchange）を解説します．これらの仕様の位置付けを**図 -1** に示します．

前回は WS-Security [1] の仕様と，セキュリティトークンサービスを介して安全なメッセージ交換を行うための仕組みの概要について説明しました．今回は，安全なメッセージ交換に必要となる 3 つの WS-SX[2] 関連仕様，WS-Trust[3]，WS-SecureConversation[4]，WS-SecurityPolicy[5] について説明します（**図 -2**）．

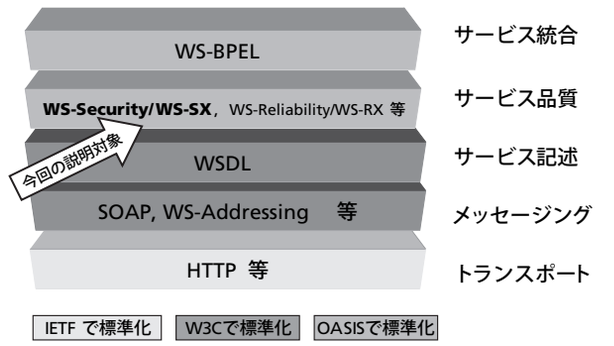


図 -1　WS-Security/WS-SX の位置付け

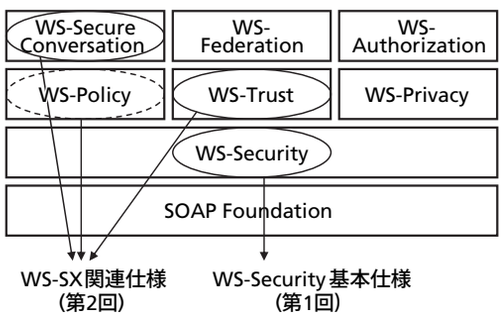


図 -2　WS-Security roadmap

セキュリティトークンサービス

セキュリティトークンサービスはセキュリティトークンの発行・検証などを行う Web サービスです．

図 -3 を見てください．旅行代理店サービスが飛行機予約サービスと通信したい場合，飛行機予約サービスが対応しているセキュリティトークンの発行をセキュリティトークンサービスに依頼します．ここで，セキュリティトークンサービスと飛行機予約サービスの間では，あらかじめ信頼関係が確立されており，飛行機予約サービスのセキュリティトークンをセキュリティトークンサービスが管理しているものとします．

セキュリティトークンサービスはセキュリティトークンの発行を依頼した旅行代理店サービスを認証した後，要求されたセキュリティトークンを発行します．このとき，発行されたセキュリティトークンに対応する鍵情報も同時に発行されます．旅行代理店は，この発行された鍵を用いて署名・暗号化したメッセージを飛行機予約サービスに送ることができます．

このようなセキュリティトークンサービスの仕様を規定しているのが WS-Trust です．WS-Trust では，セキュリティトークンサービスの WSDL や，トークン発行のためのメッセージ交換の方法やメッセージ形式が決められています．

セキュリティトークンサービスでは，旅行代理店サービスと飛行機予約サービス間でセッションが確立されている一定時間だけ有効な共有鍵と対応するセキュリティトークンを発行することもできます．このような一時的なセッション鍵を表現するセキュリティトークンを，セキュリティコンテキストトークンと呼んでいます．WS-SecureConversation は，セキュリティコンテキストトークンの形式を規定している仕様です．まずはこの 2 つの仕様について詳細を見ていきましょう．

セキュリティトークンサービスのWSDL

リスト 1 は，WS-Trust で規定されているセキュリティ

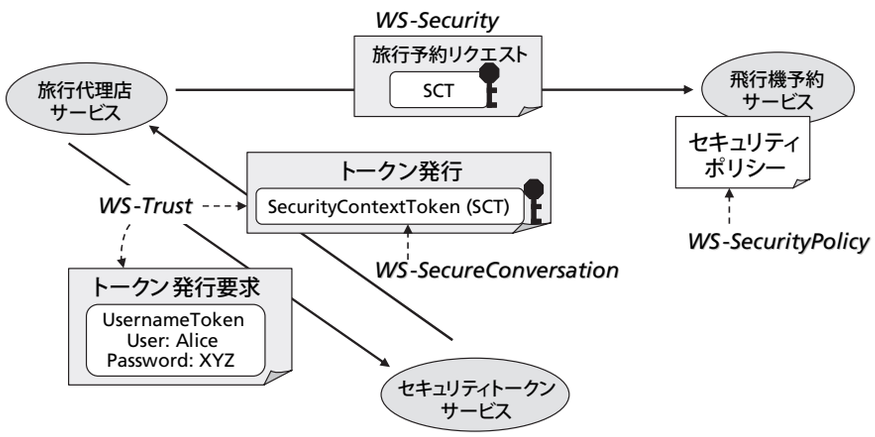


図 -3　WS-SX 関連仕様

トークンサービスの WSDL の一部です．ここでは，いくつかのメッセージ交換パターンが規定されていますが，基本となるパターンを**図 -4** に示します．

```
<!-- メッセージ形式 -->
<wsdl:message
   name="RequestSecurityTokenMsg">
  <wsdl:part name="request"
   element="wst:RequestSecurityToken" />
</wsdl:message>
<wsdl:message
   name="RequestSecurityTokenResponseMsg">
  <wsdl:part name="response"
   element="wst:RequestSecurityTokenResponse" />
</wsdl:message>
<wsdl:message
   name="RequestSecurityTokenResponseCollectionMsg">
  <wsdl:part name="responseCollection"
   element="wst:RequestSecurityTokenResponseCollection"/>
</wsdl:message>
....
<!-- セキュリティトークンサービスの portType: -->
<wsdl:portType name="SecurityTokenService">
  <wsdl:operation
   name="RequestSecurityToken">
   <wsdl:input
   message="tns:RequestSecurityTokenMsg"/>
   <wsdl:output
    message="tns:RequestSecurityTokenResponseMsg"/>
   </wsdl:operation>
  <wsdl:operation
   name="RequestSecurityToken2">
   <wsdl:input
    message="tns:RequestSecurityTokenMsg"/>
   <wsdl:output
    message="tns:RequestSecurityTokenResponseCollectionMsg"/>
  </wsdl:operation>
</wsdl:portType>
```

リスト 1．WS-Trust の WSDL の一部

リスト 1 では，セキュリティトークンサービスで交換される 3 つのメッセージが wsdl:message エレメントで定義されています．図 -4 を見てください．セキュリティトークンの要求では RequestSecurityToken 要素（以後 RST）が送られ，その結果が RequestSecurityTokenResponse 要素（以後 RSTR）により返されます．複数のメッセージ交換の後，セキュリティトークンが発行されるパターンもあります．この場合の最後のメッセージ交換では，最後の RSTR であることを示すために RequestSecurityTokenResponseCollection 要素（以後 RSTRC）の子要素として送られます．セキュリティトークンサービスには，セキュリティトークンの発行のほかにトークンの破棄・再発行・検証などの処理を要求することもできます．セキュリティトークンに対し，どのような処理を要求するかにより，具体的なメッセージ構造が決められています．

では次に，RST, RSTR のメッセージ構造を見てみましょう．

セキュリティトークン発行要求（RST）

リスト 2 はセキュリティ発行要求のための RST の例です．RST の基本的な要素は TokenType 要素と RequestType 要素の 2 つです．

TokenType 要素は，処理対象のセキュリティトークンの種類を指定します．WS-Security では，セキュリティトークンの種類ごとに TokenType を表す URI が決められており，この URI によりトークンの種類を指定します．たとえば，X509 トークンの処理を要求する場合には，http://docs.oasis-open.org/wss/2004/01/oasis-2004/01-wss-x509-token-profile-1.0#X509v3 を指定します．

RequestType 要素は，セキュリティトークンサービスに要求する処理を URI で指定します．URI は処理内容

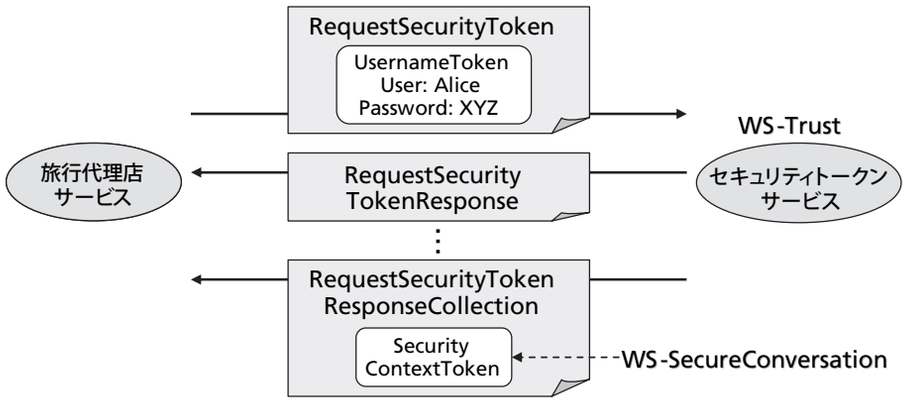
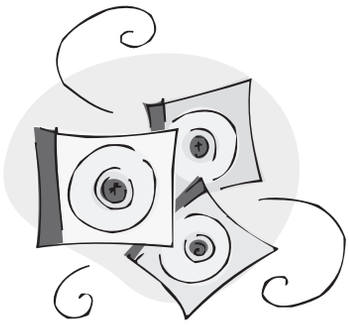


図 -4　セキュリティトークンの取得

ごとに決められており，セキュリティトークンの発行を要求する場合には http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue を指定します．

セキュリティトークンの発行要求では，TokenType 要素と RequestType 要素のほかに，追加要素が必要になります．そのひとつが wsp:AppliesTo 要素です．これは WS-PolicyAttachment[6] で定義されている要素です．ここで，どのサービスで有効なセキュリティトークンを要求するか，すなわち発行されたセキュリティトークンの有効範囲（スコープ）を指定します．

RST は Body 要素に入れられてセキュリティトークンサービスへ送られます．しかし，RST だけでは，誰がセキュリティトークンの発行を要求しているのかは分かりません．そこで，要求者の認証を行うために，要求者のセキュリティトークンが必要になります．図 -4 の例では，要求者の Username トークンが Security ヘッダに挿入されています．セキュリティトークンサービスは，この Username トークンで要求者を認証し，その後 AppliesTo で指定されているサービスへアクセスするためのセキュリティトークンを発行することになります．



```
<soap:Envelope ....>
  <soap:Header>
    <wsse:Security>
    <!-- 発行要求者の Username トークン -->
     <wsse:UsernameToken>
      <wsse:Username>
       Alice
      </wsse:Username>
      <wsse:Password
         Type="...#PasswordDigest">
       weYI3nXd8LjM......
      </wsse:Password>
     </wsse:UsernameToken>
    ...
   </wsse:Security>
  </soap:Header>
  <soap:Body>
    <wst:RequestSecurityToken>
     <!-- トークンの種類を指定 -->
     <wst:TokenType>
      http://docs.oasis-open.org/wss/2004/01/oasis-2004/
      01-wss-x509-token-profile-1.0#X509v3
     </wst:TokenType>
     <!-- トークン発行要求 -->
     <wst:RequestType>
      http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
     </wst:RequestType>
     <wsp:AppliesTo>
       http://www...../reserveAirline
     </wsp:AppliesTo>
    </wst:RequestSecurityToken>
  </soap:Body>
</soap:Envelope>
```

リスト 2．セキュリティトークン発行 RST

セキュリティトークン発行（RSTR）

リスト 3 はリスト 2 に対する返信の RSTR の例です．RequestedSecurityToken 要素は発行要求されたセキュ

リティトークンを返すための要素です．この子要素にセキュリティトークンが直接挿入されますが，セキュリティトークンへの参照を指定することもできます．ここでは，発行された X509 トークンが BinarySecurityToken 要素で挿入されています．

RequestedProofToken 要素は，発行されたセキュリティトークンに対応する鍵の情報が挿入されます．ここでは，暗号化された秘密鍵が EncryptedKey 要素の中に挿入されています．

1 つのセキュリティトークンを発行するだけでも，セキュリティトークンサービスとのメッセージ交換が複数回行われる場合もあります．その場合，最後のメッセージ交換の RSTR は，最後の RSTR であることを示すために，RSTRC に入れる必要があります．

```
<!-- 最後の RSTR は RSTRC に入る -->
<wst:RequestSecurityTokenResponseCollection>
 <wst:RequestSecurityTokenResponse>
  <!-- 発行されたセキュリティトークン -->
  <wst:RequestedSecurityToken>
   <wsse:BinarySecurityToken
    ValueType="...#X509v3"
    EncodingType="...#Base64Binary" >
     ...
   </xyz:BinarySecurityToken>
  </wst:RequestedSecurityToken>
  <!-- 対応する鍵情報 -->
  <wst:RequestedProofToken>
   <xenc:EncryptedKey ...>
    ...
   </xenc:EncryptedKey>
  </wst:RequestedProofToken>
 </wst:RequestSecurityTokenResponse>
</wst:RequestSecurityTokenResponseCollection>
```

リスト 3．セキュリティトークン発行 RSTRC

セキュリティコンテキストトークン（SCT）

2 つのサービス間で行われる安全なメッセージ交換の方法として，2 者間のセッションが確立されている間だけ有効な共有鍵を生成し，この鍵で署名・暗号化するという場合が考えられます．このような一時的に共有されるセキュリティコンテキストを表すセキュリティトークンを，セキュリティコンテキストトークン（以後 SCT）と呼びます．セキュリティトークンサービスでは，SCT を発行することもできます．

WS-SecureConversation では，SCT の表現を規定しています．リスト 4 を見てください．SCT は以下のように定義されています．

```
<wsc:SecurityContextToken ...>
  <wsc:Identifier>...</wsc:Identifier>
  <wsc:Instance>...</wsc:Instance>
   ...
</wsc:SecurityContextToken>
```

リスト 4．セキュリティコンテキストトークン（SCT）

SCT には一意の URI が割り当てられ，それを Identifier 要素に指定します．署名・暗号化において，SCT を参照する場合には，この Identifier 要素の値で参照します．SCT には，一時的に作られた共有鍵が対応しており，署名・暗号化にはこの共有鍵を使うことになります．

SCT と同時に発行された一時的な共有鍵は，セッションが終わると無効になります．しかし，同じ 2 つのサービス間で新しいセッションが開始され，ここでも一時的な共有鍵を必要とする場合，最初に発行した SCT を再発行することができます．再発行されたセキュリティトークンでは，再発行前のトークンと Identifier 要素の値は変わりませんが，対応している共有鍵そのものは変化しています．したがって，再発行前のトークンと再発行後のトークンを区別する情報が必要になります．これが Instance 要素の値になります．Instance 要素には，そのトークンに対応付けられている共有鍵ごとに一意の値が指定されます．

リスト 5，リスト 6 は WS-Trust でセキュリティコンテキストの発行を要求したときの RST，RSTR の一例です．RST では SCT に対応した TokenType 要素が指定されています．RSTR では RequestedSecurityToken 要素で SecurityContextToken 要素が返され，Identifier として UUID が使われています．RequestedProofToken 要素には，この SCT に対応付けられている共有鍵が暗号化されて入っています．

```
<wst:RequestSecurityToken>
 <wst:TokenType>
   http://docs.oasis-open.org/ws-sx/
   ws-secureconversation/200512/sct
 </wst:TokenType>
 <wst:RequestType>http://docs.oasis-open.org/ws-sx/
 ws-trust/200512/Issue</wst:RequestType>
</wst:RequestSecurityToken>
```

リスト 5. WS-Trust による SCT 発行要求

```
<wst:RequestSecurityTokenResponseCollection>
  <wst:RequestSecurityTokenResponse>
    <wst:RequestedSecurityToken>
      <wsc:SecurityContextToken>
       <wsc:Identifier>uuid:...</wsc:Identifier>
      </wsc:SecurityContextToken>
    </wst:RequestedSecurityToken>
    <wst:RequestedProofToken>
      <xenc:EncryptedKey ...>
        ...
      </xenc:EncryptedKey>
    </wst:RequestedProofToken>
  </wst:RequestSecurityTokenResponse>
</wst:RequestSecurityTokenResponseCollection>
```

リスト 6. WS-Trust による SCT 発行

WS-SecureConversation で決められている SCT は，図 -3 で示されたように WS-Trust によるセキュリティトークンサービスで生成され管理されるものであることがお分かりいただけると思います．WS-Trust と WS-SecureConversation の 2 つの仕様があわせて議論されているのは，これらが以上のような関連を持つからだと考えています．

Web サービスのセキュリティポリシー

WS-Trust や WS-SecureConversation を使った安全なセッションを確立するには，Web サービスがどのようなセキュリティトークンサービスを利用しているのか，どのようなセキュリティトークンを要求するのかなど，求める条件をポリシーとして記述し公開する必要があります．

Web サービスのポリシーの記述方法は WS-PolicyFramework で決められており，ポリシーはポリシーアサーションの組合せとして表現されます．ポリシーアサーションとは，ポリシーに記述するさまざまな要件・制約を表す XML エレメントです．リスト 7 はポリシーの構造を表しています．ポリシーアサーションは <Assertion> と表現されており，ExactlyOne 要素の子要素にポリシーアサーションの組合せが指定されます．

```
<wsp:Policy ... >
  <wsp:ExactlyOne>
    ( <wsp:All>
      ( <Assertion ...> ...
      </Assertion> )*
    </wsp:All> )*
  </wsp:ExactlyOne>
</wsp:Policy>
```

リスト 7．WS-PolicyFramework で定義されたポリシーの形式

WS-PolicyFramework に基づいて具体的なポリシーを記述するためのポリシーアサーションは，ポリシーの種類ごとに別の仕様により決められています．セキュリティの要件を記述するためのポリシーアサーションは，WS-SecurityPolicy で定義されています．

図 -5 の左側を見てください．これは WS-SecurityPolicy で書かれたセキュリティポリシーの一例です（ここでは，ExactlyOne 要素や All 要素は省略しています）．このセキュリティポリシーでは以下の要求を表しています．

「メッセージは，SOAP Header，Body を Basic256 アルゴリズムスイートのアルゴリズムにより，X509 トークンで署名・暗号化する．さらに，署名されたタイムスタンプと署名された Username トークンが必要である．」

図 -5 では，4 種類のセキュリティポリシーアサーションの組合せにより，ポリシーが記述されています．ポリシーアサーションの中には，トークンアサーションのように，他のポリシーアサーションの内部で使われるものもあります．次に，それぞれのセキュリティポリシーアサーションの詳細を見ていきましょう．

セキュリティポリシーアサーション

セキュリティバインディングアサーションは，署名・暗号化で使われるトークンやアルゴリズムなどバインディングに関する要件を表すアサーションです．図 -5 のセキュリティバインディングアサーションは，AsymmetricBinding 要素です．これは，公開鍵方式による署名・暗号化が求められていることを表します．RecipientToken 要素と InitiatorToken 要素は，署名・暗号化で使われるセキュリティトークンの種類を表す要素です．セキュリティトークンの種類は，トークンアサーションで指定します．ここではどちらも X509Token アサーションが指定されています．

署名・暗号化に使われるアルゴリズムは AlgorithmSuite 要素により指定されます．また，ここで使われるオプションがいくつか決められており，IncludeTimestamp 要素はそのひとつです．メッセージにはタイムスタンプを挿入し，さらに署名することを要求していることになります．

サポーティングトークンアサーションは，署名・暗号化以外の用途に使われる追加のセキュリティトークンを要求するためのアサーションです．ここで使われている SignedSupportingTokens 要素は，子要素に指定されたセキュリティトークンをメッセージに追加

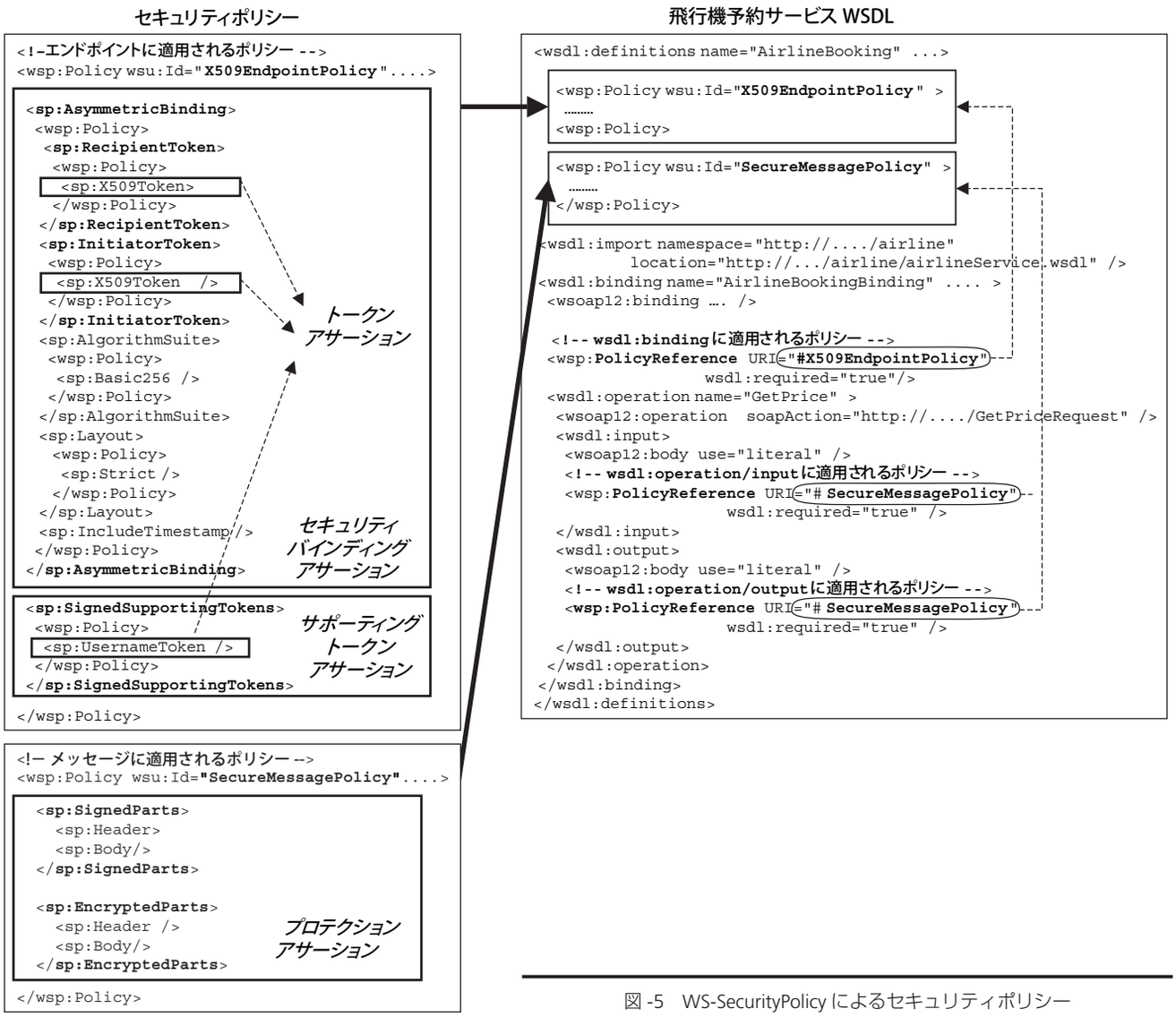


図 -5　WS-SecurityPolicy によるセキュリティポリシー

し，さらに署名することを要求しています．ここでは，UsernameToken アサーションが指定されているので，署名された UsernameToken をメッセージに挿入する必要があるということです．

プロテクションアサーションは，署名・暗号化対象を指定するためのアサーションです．SignedParts 要素では，Header, Body 要素を子要素に持つことで，SOAP Header と Body への署名が必要であることを示しています．EncryptedParts 要素も同様です．任意の場所を署名・暗号化対象に指定したい場合には，SignedElement 要素や EncryptedElement 要素により，XPath 表現で署名・暗号化対象を指定できます．

プロテクションアサーションで指定された署名・暗号化対象は，セキュリティバインディングアサーションで指定されたセキュリティトークンによる署名・暗号化が要求されていることになります．このように，プロテクションアサーションとセキュリティバインディングアサーションの組合せにより，要求されるセキュリティ要件が表現されます．しかし，セキュリティポリシーの記述上では，これら 2 つのアサーション間に明示的な関連は表現されません．セキュリティポリシーを理解するには，それぞれ独立に記述されたアサーション間に関連があるということを知っておく必要があります．このような明示的に表されない関連がほかにも複数存在します．この点が，WS-SecurityPolicy によるセキュリティポリシーを理解するうえで困難な点であるといえます．

セキュリティポリシーの適用方法

ポリシーを Web サービスに適用する方法はいくつか

ありますが, その 1 つは適用したいポリシーを Web サービスのWSDLから参照することです. このようなポリシーの適用方法については，WS-PolicyAttachment という仕様で決められています．図 -5 の右側を見てください．これは，左側のセキュリティポリシーを WSDL に適用した例です．ポリシーのアサーションは，アサーションの種類によって適用可能な対象が決まっています．ここでは，バインディングアサーションは，wsdl:binding に，プロテクションアサーションは wsdl:operation/wsdl:input と wsdl:operation/wsdl:output に適用されています．

WS-SecurityPolicy では非常に多くのアサーションが定義されており，組合せのパターンも非常に多くなっています．したがって，表現できるセキュリティ要件は非常に多く，柔軟にポリシーを設定することができます．しかし，実際にポリシーを記述するには，サービスのセキュリティ要件を正しく理解するだけでなく，ポリシーの記述方法に関する知識が要求されます．このような知識を要するということは，ポリシーの作成者にとって非常に負担が大きいと考えられます．このような理由から，今後はポリシー作成者をサポートするためのツールや手法などが必要となると考えています．

参考文献

1）Web Services Security : SOAP Message Security 1.1 (WS-Security 2004). http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
2）OASIS WS-SX TC Web ペ ー ジ. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-sx
3）WS-Trust 1.3 Committee Draft 01. http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-spec-cd-01.pdf
4）WS-SecureConversation 1.3 Committee Draft 01. http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-spec-cd-01.pdf
5）WS-SecurityPolicy 1.2 Committee Draft 02. http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-cd-02.pdf
6）WS-PolicyAttachment. http://www.ibm.com/developerworks/webservices/library/specification/ws-polatt/?S_TACT=105AGX04&S_CMP=LP

（平成 19 年 5 月 23 日受付）

佐藤史子（正会員） sfumiko@jp.ibm.com

2001 年東京工業大学理工学研究科基礎物理学専攻修了．同年日本 IBM（株）東京基礎研究所入所．現在は，Web サービスセキュリティ，サービスコンポーネントアーキテクチャのセキュリティ関連の研究に従事．