Java Web サービス: WS-Trust と WS-SecureConversation

WS-SecureConversation による Web サービス・セキュリティーのパフォーマンス向上について学ぶ

Dennis Sosnoski
Architecture Consultant and Trainer
Sosnoski Software Associates Ltd

2010年 5月 25日

WS-Security は、SOAP メッセージ交換にエンタープライズ・レベルのセキュリティー機能を追加しますが、同時にパフォーマンス面でかなりの犠牲を伴います。WS-Trust は、WS-Security をベースとしてセキュリティー・トークンを交換するための手段です。そして WS-SecureConversation は WS-Security と WS-Trust をベースに、進行中のメッセージ交換のパフォーマンスを改善します。今回の連載「Java Web サービス」では、Dennis Sosnoski が WS-Trust と WS-SecureConversation を紹介します。

このシリーズの他の記事を見る

WS-Security は、暗号方式に関して確立された業界標準と XML 暗号化および署名に関して確立された業界標準をベースに、Web サービス・アプリケーションのための包括的なセキュリティー機能を提供します。多くのアプリケーションにとって WS-Security の機能は不可欠ですが、これらのセキュリティー機能と引き換えに、パフォーマンス面でかなりの犠牲が強いられることになります。この連載の以前の記事では一般的な WS-Security 構成がパフォーマンスに与える影響を、Apache Axis2、Metro、Apache CXF という代表的な 3 つのオープンソースの Java™ Web サービス・スタックで調査しました。

WS-Security によってパフォーマンスが犠牲になる理由は、主に、非対称暗号化が幅広く使用されているためです。「Axis2 WS-Security による署名と暗号化」で説明したように、非対称暗号化は鍵のペアによって機能する便利なツールです。非対称暗号化では、ペアを構成する鍵の一方を使ってメッセージを暗号化し、もう一方の鍵を使ってメッセージを復号します。鍵のペアの所有者は、一方の鍵を公開鍵にすることで、誰でもその所有者へのメッセージをセキュアに暗号化できるようにし、所有者からのメッセージを復号できるようにします (それによって送信者の身元を確認します)。その一方、非対称暗号化の欠点は、鍵のサイズ、そして処理のオーバーヘッドです。非対称暗号化の場合、メッセージを交換する当事者だけに既知となる単一の秘密鍵を使用した単純な対称暗号化に比べ、鍵のサイズも、処理のオーバーヘッドも遥かに大きくなります。

この連載について

Web サービスは、エンタープライズ・コンピューティングにおいて Java 技術が担う重大な役割の一部です。この連載では、XML および Web サービスのコンサルタントである Dennis Sosnoski が、Web サービスを使用する Java 開発者にとって重要になる主要なフレームワークと技術について説明します。この連載から、現場での最新の開発情報を入手して、それらを皆さんのプログラミング・プロジェクトにどのように利用できるかを知っておいてください。

WS-SecureConversation は、クライアントとサーバーとの間で進行中のメッセージ交換に対称暗号化を使用できるようにすることによって、非対称暗号化がもたらすオーバーヘッドを排除する標準です。対称暗号化を使用するために必要な秘密鍵の情報をセキュアに交換できるよう、WS-SecureConversation は WS-Security と併せ、WS-Trust というもう 1 つの標準をベースとしています。WS-Trust 自体は WS-Security をベースに、セキュリティー・トークンを発行および処理する Web サービスのインターフェースを定義する標準です。

WS-Trust

WS-Trust は 2 つの関連する機能を 1 つに結合した標準です。一方の機能はセキュリティー・トークン処理のサポートで、具体的にはセキュリティー・トークンの発行、更新、取り消しをサポートします。もう一方の機能は、信頼関係の仲介をサポートすることです。この 2 つは異なる機能のように思えるかもしれませんが、セキュリティー・トークンは信頼される必要があり、信頼はトークンという形で表す必要があるという点で、相互に関係します。

WS-Trust の中核となるのは、セキュリティー・トークンの発行、更新、取り消し、そして検証を行うためのメッセージ一式です。クライアントは STS (Security Token Service) という特定のタイプの SOAP Web サービスを呼び出すことで、これらのメッセージを交換することができます。また、STS 以外の方法でもメッセージを渡すことができます (他のサービスに対するリクエストのセキュリティー・ヘッダーに含めるなど)。

STS の基本

STS は、WS-Trust 仕様で定義されている単純なインターフェースを実装する Web サービスです。このインターフェースでは、クライアントがセキュリティー・トークンを使って数種類の操作に対するリクエストを送信することができます。STS は Web サービスであることから、リクエスト・メッセージとレスポンス・メッセージで直接 WS-Security を使用することも、WS-SecurityPolicy を使って、クライアントが提供するクレデンシャルのタイプや、メッセージで必要となる別のセキュリティー処理を指定することもできます。

最も基本的な操作は、新しいトークンを発行する際の操作です。リスト 1 に、STS にトークンの発行を求めるリクエストの例を記載します。ここではリクエストを編集して各種のヘッダーを取り除き、リクエスト本体だけを記載しています (ヘッダーが含まれる例は、後で記載します)。リスト 1 のリクエストで要求しているのは、WS-SecureConversation が使用する SCT (Security Context Token) という特定の種類のトークンです。

リスト 1. STS に対するセキュリティー・トークンのリクエスト

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    ...
  </soap:Header>
  <soap:Body xmlns:wsu=".../oasis-200401-wss-wssecurity-utility-1.0.xsd"
```

```
      wsu:Id="Id-7059772">
    <wst:RequestSecurityToken
        xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
      <wst:RequestType>http://.../ws-sx/ws-trust/200512/Issue</wst:RequestType>
      <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
        <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">
          <wsa:Address>http://localhost:8800/cxf-seismicsc-signencr/</wsa:Address>
        </wsa:EndpointReference>
      </wsp:AppliesTo>
      <wst:Lifetime xmlns:wsu=".../oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <wsu:Created>2010-05-12T10:33:22.774Z</wsu:Created>
        <wsu:Expires>2010-05-12T10:38:22.774Z</wsu:Expires>
      </wst:Lifetime>
      <wst:TokenType
      >http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct</wst:TokenType>
      <wst:KeySize>128</wst:KeySize>
      <wst:Entropy>
        <wst:BinarySecret Type="http://docs.oasis-open.org/ws-sx/ws-trust/200512/Nonce"
        >kIYFB/u430k3PlOPfUtJ5A==</wst:BinarySecret>
      </wst:Entropy>
      <wst:ComputedKeyAlgorithm
      >http://.../ws-sx/ws-trust/200512/CK/PSHA1</wst:ComputedKeyAlgorithm>
    </wst:RequestSecurityToken>
  </soap:Body>
</soap:Envelope>
```

リスト 1 のリクエスト本体には、STS へのリクエストのほとんどで使用する基本的な `<wst:RequestSecurityToken>` 要素が示されています。`<wst:RequestType>` はリクエストのタイプを識別する、この要素に必須の子要素です (この例では、`Issue` リクエストとなっています)。残りの子要素は、以下の内容を指定する `Issue` リクエストのオプション・パラメーターです。

- 要求するトークンを使用してアクセスするサービス・エンドポイント (`<wsp:AppliesTo>` 要素)
- トークンの有効期間 (`<wst:Lifetime>` 要素)
- トークンのタイプ (`<wst:TokenType>` 要素)
- 要求する鍵のサイズ (ビット単位) (`<wst:KeySize>` 要素)
- 秘密鍵を生成する際に使用する、クライアント提供のエントロピー・データ (`<wst:Entropy>` 要素)
- 秘密鍵を生成する際に使用するアルゴリズム (`<wst:ComputedKeyAlgorithm>` 要素)

リクエストを受信した STS が、クライアントから必須情報として提供されたクレデンシャルを承認し、リクエストの条件に同意した場合には、`Issue` リクエストに対するレスポンスにセキュリティー・トークンを含めて返します。リスト 2 に、`Issue` リクエストに対する正常なレスポンスの例を記載します。この例でも、ヘッダーは省略してあります。

リスト 2. STS から返されたセキュリティー・トークンが含まれるレスポンス

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    ...
  </soap:Header>
  <soap:Body xmlns:wsu=".../oasis-200401-wss-wssecurity-utility-1.0.xsd"
      wsu:Id="Id-4824957">
    <wst:RequestSecurityTokenResponseCollection
        xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
      <wst:RequestSecurityTokenResponse>
        <wst:RequestedSecurityToken>
          <wsc:SecurityContextToken
              xmlns:wsc="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"
```

```
              xmlns:wsu=".../oasis-200401-wss-wssecurity-utility-1.0.xsd"
              wsu:Id="sctId-A167EB2B526E0894DA12736604029099">
            <wsc:Identifier>A167EB2B526E0894DA12736604029098</wsc:Identifier>
          </wsc:SecurityContextToken>
        </wst:RequestedSecurityToken>
        <wst:RequestedAttachedReference>
          <wsse:SecurityTokenReference
            xmlns:wsse=".../oasis-200401-wss-wssecurity-secext-1.0.xsd">
            <wsse:Reference xmlns:wsse=".../oasis-200401-wss-wssecurity-secext-1.0.xsd"
                URI="#sctId-A167EB2B526E0894DA12736604029099"
                ValueType=".../ws-sx/ws-secureconversation/200512/sct"/>
          </wsse:SecurityTokenReference>
        </wst:RequestedAttachedReference>
        <wst:RequestedUnattachedReference>
          <wsse:SecurityTokenReference
            xmlns:wsse=".../oasis-200401-wss-wssecurity-secext-1.0.xsd">
            <wsse:Reference xmlns:wsse=".../oasis-200401-wss-wssecurity-secext-1.0.xsd"
            URI="A167EB2B526E0894DA12736604029098"
            ValueType=".../ws-sx/ws-secureconversation/200512/sct"/>
          </wsse:SecurityTokenReference>
        </wst:RequestedUnattachedReference>
        <wst:Lifetime xmlns:wsu=".../oasis-200401-wss-wssecurity-utility-1.0.xsd">
          <wsu:Created>2010-05-12T10:33:22.909Z</wsu:Created>
          <wsu:Expires>2010-05-12T10:38:22.909Z</wsu:Expires>
        </wst:Lifetime>
        <wst:RequestedProofToken>
          <wst:ComputedKey
          >http://docs.oasis-open.org/ws-sx/ws-trust/200512/CK/PSHA1</wst:ComputedKey>
        </wst:RequestedProofToken>
        <wst:Entropy>
          <wst:BinarySecret Type="http://docs.oasis-open.org/ws-sx/ws-trust/200512/Nonce"
          >DpkK6qcELTO8dlPdDHMi2A==</wst:BinarySecret>
        </wst:Entropy>
      </wst:RequestSecurityTokenResponse>
    </wst:RequestSecurityTokenResponseCollection>
  </soap:Body>
</soap:Envelope>
```

リスト 2 に記載されたレスポンスには、`<wst:RequestSecurityTokenResponseCollection>` 要素の中に `<wst:RequestSecurityTokenResponse>` 要素が 1 つあり、この要素の中にレスポンス・トークンの情報がラップされています。このリクエストが要求しているのは SCT という特定のトークンで、リスト 1 のリクエスト・メッセージでは以下の値として示されています。

```
<wst:TokenType>http://schemas.xmlsoap.org/ws/2005/02/sc/sct</wst:TokenType>
```

そのため、上記のレスポンスには以下の内容が含まれます。

- 実際の SCT (`<wst:RequestedSecurityToken>` 要素内にラップ)
- 複数のリファレンス構造 (`<wst:RequestedAttachedReference>` および `<wst:RequestedUnattachedReference>` 要素)
- トークンの有効期間 (`<wst:Lifetime>` 要素)
- 証明トークン (`<wst:RequestedProofToken>` 要素)
- 秘密鍵を生成する際に使用する、サーバー提供のエントロピー・データ (`<wst:Entropy>` 要素)

証明トークンの中で指定される内容は、対称暗号化をする際に基本の値として使用する共有シークレットの値です。この例における共有シークレットの値は、クライアントとサーバーがそれ

ぞれ提供するエントロピーの値を組み合わせ、指定されたアルゴリズムを使って生成されています。

その他のオプション

`Issue` リクエスト・タイプの他に STS に対して要求できるリクエストには、`Validate`、`Renew`、および `Cancel` があります。これらのリクエストではいずれも、前に発行されたトークンを参照するか、もしくは提供しなければなりません。それによって、クライアントはトークンを検証したり、トークンの有効期間を延長または終了したりすることができます。

STS がレスポンスで 1 つのトークンだけを返す場合、リスト 2 のように `<wst:RequestSecurityTokenCollection>` 要素にトークンをラップする代わりに、レスポンスで `<wst:RequestSecurityTokenResponse>` 要素を直接使用することもできます。STS に対するリクエストには、任意の数の `<wst:RequestSecurityToken>` 要素をラップする `<wst:RequestSecurityTokenCollection>` 要素を使用することができます。

WS-Trust では、STS Web サービス・インターフェースを使用せずに、SOAP メッセージ・ヘッダーに直接セキュリティー・トークンを含めることもできるようになっています。サービスと同じ場所に配置されていない STS からトークンを取得する場合には、これが、トークンを共有する唯一の手段となります。

WS-SecureConversation

WS-Trustをベースとする WS-SecureConversation では、STS を使用して SCT を管理します。SCT が表すのは、メッセージ交換の当事者間で共有されるコンテキストです。この共有コンテキストの情報を使用することで、メッセージの送信側と受信側は対称暗号化を使ってメッセージをセキュアにすることができます。

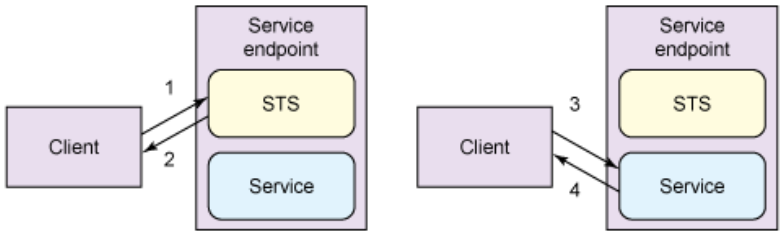
このコンテキストが具体的に提供するのは、メッセージの送信側と受信側の間での共有シークレットの値です (リスト 2 のレスポンス・メッセージを参照)。共有シークレット自体を秘密鍵として使用して、交換するメッセージの対称暗号化を行うこともできますが、それよりも望ましい方法は、共有シークレットを基本の値として使用し、メッセージ交換で使用される実際の秘密鍵を導き出すことです。無駄に複雑にしているように聞こえるかもしれませんが、この方法は、メッセージ交換をモニターして秘密鍵を解明しにくくすることを目的としています。

STS とサービス

理論上、WS-SecureConversation はマルチパーティー・メッセージ交換で使用できますが、最も一般的な使用方法は、単一のサーバーと通信するクライアントに使用することです。この構成で使用する場合、SCT をクライアントに提供する STS はサーバーと同じ場所にあり、サーバーと同じエンドポイント・アドレスでアクセスすることになります。これはつまり、サーバー上の Web サービスのコードには、STS を対象とするメッセージとサービス自体を対象とするメッセージとを区別する手段が必要であることを意味します。その手段となるのが、リクエストで使用されるアクションです。

図 1 に、サービスと同じ場所にある STS の構成およびメッセージ交換を図解します。

図 1. サービスと同じ場所に配置された WS-SecureConversation STS



クライアントがサーバーとのメッセージ交換を開始するときには、まず STS に連絡してコンテキストを設定します。このメッセージ (図 1 に「1」として示したメッセージ) は、アクション `http://schemas.xmlsoap.org/ws/2005/02/trust/RST/SCT` を指定します。それに対するレスポンス (メッセージ「2」) は、この SCT をクライアントに送信します。このコンテキストがメッセージ「3」で参照されることによって、実際のサービス・アプリケーションに関連付けるアクションが指定されます。この SCT はその有効期間中、クライアントからサービスに送信される、それ以降のすべてのメッセージで有効です。メッセージ「3」とメッセージ「4」は、共有シークレットに基づく対称暗号化を使用します。これは、クライアントとサービスとの間でのそれ以降のすべてのメッセージでも同様です。サービス・アプリケーションは、クライアントが提供するコンテキスト参照を使用して、STS が保持するコンテキストによる共有シークレットに直接アクセスします。

WS-Policy 構成

WS-SecureConversation が使用する WS-Policy 構成と WS-SecurityPolicy 構成は、連載の以前の記事で説明した基本的な WS-Security 処理で使用する構成と似ています。ただし、WS-SecureConversation を使用する場合には、ポリシーが 2 つの別個のメッセージ交換をサポートしなければならないという点が大きく異なります。2 つのメッセージ交換とは、クライアントと STS との間でのメッセージ交換、そしてクライアントと実際のサービスとの間でのメッセージ交換です。これらをサポートするポリシー記述で、STS とのメッセージ交換にはネストされたポリシーを使用し、クライアントとサービスとの間でのメッセージ交換はポリシー本体を適用することによって対処します。

リスト 3 に、この記事の例で使用するポリシーを記載します。

リスト 3. WS-SecureConversation ポリシーの例

```
<wsp:Policy wsu:Id="SecConv"
    xmlns:wsu=".../oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
  <wsp:ExactlyOne>
    <wsp:All>
      <wsap:UsingAddressing xmlns:wsap="http://www.w3.org/2006/05/addressing/wsdl"/>
      <sp:SymmetricBinding>
        <wsp:Policy>
          <sp:ProtectionToken>
            <wsp:Policy>
              <sp:SecureConversationToken sp:IncludeToken=".../AlwaysToRecipient">
                <wsp:Policy>
                  <sp:RequireDerivedKeys/>
                  <sp:BootstrapPolicy>
                    <wsp:Policy>
                      <sp:AsymmetricBinding>
```

```
                    <wsp:Policy>
                      <sp:InitiatorToken>
                        <wsp:Policy>
                          <sp:X509Token sp:IncludeToken=".../AlwaysToRecipient">
                            <wsp:Policy>
                              <sp:RequireThumbprintReference/>
                            </wsp:Policy>
                          </sp:X509Token>
                        </wsp:Policy>
                      </sp:InitiatorToken>
                      <sp:RecipientToken>
                        <wsp:Policy>
                          <sp:X509Token sp:IncludeToken=".../IncludeToken/Never">
                            <wsp:Policy>
                              <sp:RequireThumbprintReference/>
                            </wsp:Policy>
                          </sp:X509Token>
                        </wsp:Policy>
                      </sp:RecipientToken>
                      <sp:AlgorithmSuite>
                        <wsp:Policy>
                          <sp:TripleDesRsa15/>
                        </wsp:Policy>
                      </sp:AlgorithmSuite>
                      <sp:IncludeTimestamp/>
                      <sp:OnlySignEntireHeadersAndBody/>
                    </wsp:Policy>
                  </sp:AsymmetricBinding>
                  <sp:SignedParts>
                    <sp:Body/>
                    <sp:Header Name="To" Namespace=".../addressing"/>
                    <sp:Header Name="From" Namespace=".../addressing"/>
                    <sp:Header Name="FaultTo" Namespace=".../addressing"/>
                    <sp:Header Name="ReplyTo" Namespace=".../addressing"/>
                    <sp:Header Name="MessageID" Namespace=".../addressing"/>
                    <sp:Header Name="RelatesTo" Namespace=".../addressing"/>
                    <sp:Header Name="Action" Namespace=".../addressing"/>
                  </sp:SignedParts>
                  <sp:Trust13>
                    <wsp:Policy>
                      <sp:MustSupportIssuedTokens/>
                      <sp:RequireClientEntropy/>
                      <sp:RequireServerEntropy/>
                    </wsp:Policy>
                  </sp:Trust13>
                </wsp:Policy>
              </sp:BootstrapPolicy>
            </wsp:Policy>
          </sp:SecureConversationToken>
        </wsp:Policy>
      </sp:ProtectionToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          <sp:Basic128Rsa15/>
        </wsp:Policy>
      </sp:AlgorithmSuite>
    </wsp:Policy>
  </sp:SymmetricBinding>
  <sp:EncryptedParts>
    <sp:Body/>
  </sp:EncryptedParts>
 </wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>
```

リスト 3 では、外側のポリシーは対称暗号化 (<sp:SymmetricBinding>) を使用して、交換するメッセージの本体を暗号化するように指定しています (リストの終わり近くにある <sp:EncryptedParts> の設定)。対称暗号化ポリシーの内側にある <sp:ProtectionToken> 要素と、そこにネストされた <sp:SecureConversationToken> 要素は、WS-SecureConversation を使用して対称暗号化を実行するように指定しています。

STS にアクセスするときに適用されるポリシーは、<sp:SecureConversationToken> 内にネストされた <sp:BootstrapPolicy> (太字で記載) によって定義されます。このポリシーは、X.509 証明書を使用してメッセージ本体とアドレス指定ヘッダーに署名を付けることを指定しているだけにすぎません。この署名は、連載の WS-Security に関する以前の記事に記載したのと同じタイプの署名です。

このポリシーが使用される場合、クライアントと STS との間のメッセージ交換は暗号化されないことに注意してください。これはメッセージ交換の内容を理解しやすくするためですが、実際には TLS/SSL トランスポート暗号化または WS-Security 暗号化のいずれかを使用してメッセージ交換をセキュアにする必要があります。

メッセージ交換

リスト 4 に、メッセージ「1」(STS へのリクエスト) とメッセージ「2」(クライアントへのレスポンス) のヘッダーを記載します (この 2 つのメッセージの本体は、それぞれリスト 1 とリスト 2 に記載されているとおりです)。

リスト 4. STS リクエストおよびレスポンスのヘッダー

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>

    <Action xmlns="http://www.w3.org/2005/08/addressing"
      xmlns:wsu="...wssecurity-utility-1.0.xsd" wsu:Id="Id-32320445"
      >http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT</Action>
    <MessageID xmlns="http://www.w3.org/2005/08/addressing"
      xmlns:wsu="...wssecurity-utility-1.0.xsd" wsu:Id="Id-2673180"
      >urn:uuid:24ce01d5-3c17-4df6-ad89-2fc0720152cd</MessageID>
    <To xmlns="http://www.w3.org/2005/08/addressing"
      xmlns:wsu="...wssecurity-utility-1.0.xsd" wsu:Id="Id-5132526"
      >http://localhost:8800/cxf-seismicsc-signencr/</To>
    ...
    <wsse:Security xmlns:wsse="...wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">
      <wsse:BinarySecurityToken xmlns:wsse="...wssecurity-secext-1.0.xsd"
        xmlns:wsu="...wssecurity-utility-1.0.xsd"
        EncodingType="...soap-message-security-1.0#Base64Binary"
        ValueType="...x509-token-profile-1.0#X509v3"
        wsu:Id="CertId-CF15C330C32618BF4912736604028486"
        >MIICo...8/0n33w==</wsse:BinarySecurityToken>
      <wsu:Timestamp xmlns:wsu="...wssecurity-utility-1.0.xsd" wsu:Id="Timestamp-7">
        <wsu:Created>2010-05-12T10:33:22.831Z</wsu:Created>
        <wsu:Expires>2010-05-12T10:38:22.831Z</wsu:Expires>
      </wsu:Timestamp>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="Signature-8">
        <ds:SignedInfo>
          ...
          <ds:Reference URI="#Id-7059772">
            ...
          </ds:Reference>
          ...
          <ds:Reference URI="#Timestamp-7">
            ...
```

```
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>TYIbt...V0dd8=</ds:SignatureValue>
        <ds:KeyInfo Id="KeyId-CF15C330C32618BF4912736604028487">
          <wsse:SecurityTokenReference xmlns:wsse="...wssecurity-secext-1.0.xsd"
            xmlns:wsu="...wssecurity-utility-1.0.xsd"
            wsu:Id="STRId-CF15C330C32618BF4912736604028488">
            <wsse:Reference xmlns:wsse="...wssecurity-secext-1.0.xsd"
            URI="#CertId-CF15C330C32618BF4912736604028486"
            ValueType="...x509-token-profile-1.0#X509v3"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body xmlns:wsu="..." wsu:Id="Id-7059772">
    ...
  </soap:Body>
</soap:Envelope>

soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Action xmlns="http://www.w3.org/2005/08/addressing"
      xmlns:wsu="...wssecurity-utility-1.0.xsd" wsu:Id="Id-33522601"
      >http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/SCT</Action>
    <MessageID xmlns="http://www.w3.org/2005/08/addressing"
      xmlns:wsu="...wssecurity-utility-1.0.xsd" wsu:Id="Id-9229531"
      >urn:uuid:d9d1b9b2-a864-446b-ab81-3176f868046e</MessageID>
    <To xmlns="http://www.w3.org/2005/08/addressing"
      xmlns:wsu="...wssecurity-utility-1.0.xsd" wsu:Id="Id-25551189"
      >http://www.w3.org/2005/08/addressing/anonymous</To>
    <RelatesTo xmlns="http://www.w3.org/2005/08/addressing"
      xmlns:wsu="...wssecurity-utility-1.0.xsd" wsu:Id="Id-32148925"
      >urn:uuid:24ce01d5-3c17-4df6-ad89-2fc0720152cd</RelatesTo>
    <wsse:Security xmlns:wsse="...wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">
      <wsu:Timestamp xmlns:wsu=
        "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wssecurity-utility-1.0.xsd"
        wsu:Id="Timestamp-7">
        <wsu:Created>2010-05-12T10:33:22.913Z</wsu:Created>
        <wsu:Expires>2010-05-12T10:38:22.913Z</wsu:Expires>
      </wsu:Timestamp>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="Signature-8">
        <ds:SignedInfo>
          ...
          <ds:Reference URI="#Id-4824957">
            ...
          </ds:Reference>
          ...
          <ds:Reference URI="#Timestamp-7">
            ...
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>tr1tx...GY4wk=</ds:SignatureValue>
        <ds:KeyInfo Id="KeyId-A167EB2B526E0894DA127366040291811">
          <wsse:SecurityTokenReference xmlns:wsse="...wssecurity-secext-1.0.xsd"
            xmlns:wsu="...wssecurity-utility-1.0.xsd"
            wsu:Id="STRId-A167EB2B526E0894DA127366040291812">
            <wsse:KeyIdentifier EncodingType="...soap-message-security-1.0#Base64Binary"
            ValueType="...soap-message-security-1.1#ThumbprintSHA1"
            >uYn3PK2wXheN2lLZr4n2mJjoWE0=</wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body xmlns:wsu="...wssecurity-utility-1.0.xsd" wsu:Id="Id-4824957">
    ...
```

```
  </soap:Body>
</soap:Envelope>
```

リスト 4 を見ると、クライアントからサーバーに証明書が送信され、クライアントには証明書参照が返されることがわかります。そして、それぞれの方向で送信される証明書を使って、タイムスタンプとメッセージ本体の署名が検証されます。このポリシー構成では、STS がクライアント証明書を信頼すること、そしてクライアントのトラスト・ストアに STS 証明書が存在することが必要です。

リスト 5 に、WS-SecureConversation を使用してクライアントとサービスとの間で行われるメッセージ交換を示します (かなり編集されています)。

リスト 5. サービスへのリクエストとクライアントへのレスポンス

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Action xmlns="http://www.w3.org/2005/08/addressing">urn:matchQuakes</Action>
    <MessageID xmlns="http://www.w3.org/2005/08/addressing"
      >urn:uuid:c724a446-4375-4e8a-a318-fd3c84510eae</MessageID>
    ...
    <wsse:Security xmlns:wsse="...wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">
      <wsc:SecurityContextToken xmlns:wsc=".../ws-secureconversation/200512"
        xmlns:wsu="...wssecurity-utility-1.0.xsd"
        wsu:Id="sctId-A167EB2B526E0894DA12736604029099">
        <wsc:Identifier>A167EB2B526E0894DA12736604029098</wsc:Identifier>
      </wsc:SecurityContextToken>
      <wsc:DerivedKeyToken xmlns:wsc=".../ws-secureconversation/200512"
        xmlns:wsu="...wssecurity-utility-1.0.xsd" wsu:Id="derivedKeyId-9">
        <wsse:SecurityTokenReference xmlns:wsse="...wssecurity-secext-1.0.xsd">
          <wsse:Reference xmlns:wsse="..." URI="#sctId-A167EB2B526E0894DA12736604029099"
          ValueType="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct"/>
        </wsse:SecurityTokenReference>
        <wsc:Offset>0</wsc:Offset>
        <wsc:Length>16</wsc:Length>
        <wsc:Nonce>AyUGKYBNNQstD9EmZUJqlA==</wsc:Nonce>
      </wsc:DerivedKeyToken>
      <wsc:DerivedKeyToken xmlns:wsc=".../ws-secureconversation/200512"
        xmlns:wsu="...wssecurity-utility-1.0.xsd" wsu:Id="derivedKeyId-11">
        ...
      </wsc:DerivedKeyToken>
      <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
        <xenc:DataReference URI="#EncDataId-12"/>
      </xenc:ReferenceList>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="Signature-10">
        <ds:SignedInfo>
          ...
          <ds:Reference URI="#Id-28812627">
            ...
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>6NHo8Si1ntZIb2Ivg3S/n1+2uzI=</ds:SignatureValue>
        <ds:KeyInfo Id="KeyId-CF15C330C32618BF4912736604029689">
          <wsse:SecurityTokenReference xmlns:wsse="..." xmlns:wsu="..."
          wsu:Id="STRId-CF15C330C32618BF49127366040296810">
            <wsse:Reference xmlns:wsse="..." URI="#derivedKeyId-9"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body xmlns:wsu="...wssecurity-utility-1.0.xsd" wsu:Id="Id-28812627">
    <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" ...>
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
```

```
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <wsse:SecurityTokenReference xmlns:wsse="...wssecurity-secext-1.0.xsd">
          <wsse:Reference xmlns:wsse="..." URI="#derivedKeyId-11"/>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>+krS8lGA...CKSN0fwKR36Q==</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </soap:Body>
</soap:Envelope>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Action xmlns="http://www.w3.org/2005/08/addressing"
      >http://ws.sosnoski.com/seismic/wsdl/SeismicInterface/quakeResponse</Action>
    <MessageID xmlns="http://www.w3.org/2005/08/addressing"
      >urn:uuid:c3aa0671-8751-4d6b-8d4c-0e37ce3e394a</MessageID>
    <To xmlns="http://www.w3.org/2005/08/addressing"
      >http://www.w3.org/2005/08/addressing/anonymous</To>
    <RelatesTo xmlns="http://www.w3.org/2005/08/addressing"
      >urn:uuid:c724a446-4375-4e8a-a318-fd3c84510eae</RelatesTo>
    <wsse:Security xmlns:wsse="...wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">
      <wsc:DerivedKeyToken xmlns:wsc="...ws-secureconversation/200512"
      ...
      </wsc:DerivedKeyToken>
      <wsc:DerivedKeyToken xmlns:wsc="...ws-secureconversation/200512"
      ...
      </wsc:DerivedKeyToken>
      <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
        <xenc:DataReference URI="#EncDataId-12"/>
      </xenc:ReferenceList>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="Signature-10">
        <ds:SignedInfo>
          ...
          <ds:Reference URI="#Id-10766816">
            ...
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>rU6YoV7BiO0qSQjWw2vwCp9R+fg=</ds:SignatureValue>
        <ds:KeyInfo Id="KeyId-A167EB2B526E0894DA127366040304813">
          <wsse:SecurityTokenReference xmlns:wsse="...wssecurity-secext-1.0.xsd" ...>
            <wsse:Reference xmlns:wsse="..." URI="#derivedKeyId-9"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body xmlns:wsu="...wssecurity-utility-1.0.xsd" wsu:Id="Id-10766816">
    <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" ...>
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <wsse:SecurityTokenReference xmlns:wsse="...wssecurity-secext-1.0.xsd">
          <wsse:Reference xmlns:wsse="..." URI="#derivedKeyId-11"/>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>Cl0iUu...TJ6WkZl2A==</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </soap:Body>
</soap:Envelope>
```

リスト 5 では、各メッセージのヘッダーに組み込まれた `SecurityContextToken` が、`<wsc:DerivedKeyToken>` 要素によって参照されています。この要素が指定するパラメーターによって、データの署名と暗号化で実際に使用される秘密鍵が導き出されます。

どれだけの効果があるのでしょうか

今回の記事で WS-Trust と WS-SecureConversation の基礎を学んだところで、連載の次回の記事では、WS-SecureConversation が Apache Axis2、Metro、そして Apache CXF の各 Web サービス・スタックでもたらすパフォーマンス・ゲインを調べます。パフォーマンスの結果と併せて、この 3 つのスタックでの WS-SecureConversation 構成の詳細についても説明します。

著者について

Dennis Sosnoski

Dennis Sosnoski は Java ベースの XML および Web サービスを専門とするコンサルタント兼トレーナーです。専門家としてのソフトウェア開発経験は 30 年以上に渡り、この 10 年間はサーバー・サイドの XML 技術や Java 技術に注力しています。オープンソースのJiBX XML Data Binding フレームワークや、それに関連した JiBX/WS Web サービス・フレームワークの開発リーダーを務め、さらに Apache Axis2 Web サービス・フレームワークのコミッターでもあります。彼は JAX-WS 2.0 および JAXB 2.0 仕様のエキスパート・グループの一員でもありました。