SOAP security extensions: digital signature

SOAP-DSIG and SSL

Satoshi Hada (satoshi@jp.ibm.com)
Sr. Engineer, IBM Tokyo Research Laboratory
01 Aug 2001

SOAP Security Extensions: Digital Signature (SOAP-DSIG) defines the syntax and processing rules for digitally signing SOAP messages and validating signatures. This article discusses how SOAP-DSIG is related to SSL, and describes how the two technologies complement each other.

Digital signatures allow messages to be as authentically sent by the origin user or software. Unfortunately, the Simple Object Access Protocol (SOAP) 1.1 did not include provisions for signing messages and thus lacks this security. My cohorts and I sent a proposal for adding digital signature technology to SOAP, which has since been accepted by the World Wide Web Consortium as the SOAP-DSIG Note, defining the syntax and processing rules for digitally signing SOAP messages and validating signatures. This technology has since been implemented into shipping products from IBM, Microsoft and others.

SOAP-DSIG, however, has to work with Secure Sockets Layer (SSL), the most popular and widely deployed security technology used by Web sites. Therefore we should raise the questions: How is SOAP-DSIG related to SSL? And what are the differences between these two technologies?

This article addresses these questions and describes how these two technologies provide complementary functions in areas where the other is lacking. Also, since HTTP (that is SOAP over HTTP) is so widely used, this article focuses primarily on HTTP as the transport layer. However, you should note that both SOAP and SOAP-DSIG are transport-independent and can be used over other transport protocols such as SMTP, FTP and MQ. When using other transport protocols, you need to know how SOAP-DSIG is related to the corresponding security of that transport layer (e.g. S/MIME in SMTP), as I will explain later in this article.

Introduction
Although HTTP was originally developed as a protocol to simply transfer HTML documents, now you can use it to order products and services by using CGI scripts or Java Servlets on a Web site. When ordering products over the Internet, you may need to send personal information such as credit card numbers. However, you should only send this information to trustworthy HTTP servers, and in a secure way so that no adversarial third-party can intercept and steal such information. SSL was developed to address these problems of confidentiality and server authentication, and is now widely used .

In contrast to these Business-to-Customer (B2C) applications, in Business-to-Business (B2B) applications, instead of a human being using a browser to display HTML documents, a computer processes the order. And data such as a merchandise order may be described in XML rather than HTML, and exchanged over HTTP and SOAP.

SOAP is a standard messaging layer used to exchange any XML documents and one of the main building blocks for Web services. In addition to SOAP there are related technologies such as Universal Description, Discovery,
and Integration (UDDI) and Web Service Description Language (WSDL), but this article does not address these technologies. (For links to these and other technologies mentioned in this article, please see the Resources section.)

When developing Web services and B2B applications based on SOAP, security issues are still important. In particular, in business transactions among enterprises, the security requirement of non-repudiation needs to be satisfied. SOAP-DSIG is being proposed for this purpose. This article answers the following questions: What is non-repudiation? How are SOAP-DSIG and SSL combined in order to realize it?

Security requirements for message transmission

Every SOAP message has a SOAP envelope and SOAP encoding. The SOAP envelope is a data structure that can be used to carry any XML document. SOAP encoding is used to encode non-XML data into an XML document, so that it can be transmitted within a SOAP envelope. Typically, this encoding is intended to be used in an application like a remote procedure call (RPC). Because the discussion in this article focuses on the SOAP envelope and is not directly concerned with SOAP encoding, it applies to any kind of SOAP-based application, including RPC and B2B applications.

I'll begin by summarizing the general security requirements for message transmission from one computer (sender) to another computer (recipient). Specifically, I'll consider message authentication, sender/recipient authentication, and non-repudiation. Note that the security requirements described here are not specific to SOAP, but they apply to any kind of message transmission.

The first requirement is encryption for confidentiality. Because the confidentiality requirement is satisfied by using SSL and is not addressed by SOAP-DSIG, I will not cover it in this article. The security requirement I am concerned with is authentication. Consider the following two issues:

- From the sender's perspective: When sending a message, how is the identity of the intended recipient authenticated?
- From the recipient's perspective: When receiving a message, how is the identity of the sender and the content of the message authenticated?

Here, I'll consider authentication as involving two kinds of security requirements. One is authentication of the identity of the creator of the message, which is called message authentication. The other is authentication of the identities of the sender and of the recipient, which is called sender/recipient authentication. In network environments where there may be unreliable or malicious computers, the creator of a message is not always the same as the sender of the message. For example, once a malicious party has somehow stolen a legitimate message created by a sender, the message could be forwarded to anyone. Therefore, this distinction is important.

The two types of authentication involve the following aspects:

- **Message authentication:**
  Message authentication guarantees that a transmitted message was not modified en route and the identity of the creator is not misrepresented. Typically, message authentication can be done by appending a digital signature or Message Authentication Code (MAC) to the transmitted message. What you should note here is that message authentication guarantees nothing about who sent that message.

- **Sender/recipient authentication:**
Sender and recipient authentication guarantees that a sender and a recipient are who they say they are, respectively. That is, a sender can confirm the identity of the recipient who the sender wants to send a message to, and a recipient can confirm the identity of the sender from whom a message was received. Note that sender/recipient authentication guarantees nothing about who created that message.

In the following sections, I'll summarize the security technologies that realize the above two security requirements.

Technologies for message authentication
As mentioned above, there are two common technologies used to satisfy the requirement of message authentication: the Message Authentication Code and the digital signature. Here are some of their merits and demerits.

- **Message Authentication Code (MAC):**
  SSL appends MACs to the transmitted messages and SOAP-DSIG can also be used to append MACs. Because a MAC can be computed faster than a digital signature, it is practical for transport layer security like SSL, where a large volume of data is transmitted. However, because MAC is computed by a secret key shared between the sender and the recipient, it is only guaranteed that the transmitted messages are created by either the sender or the recipient. In other words, from the perspective of some third party, you cannot determine whether the messages were created by the sender or the recipient.

- **Digital signature:**
  The original motivation for SOAP-DSIG was to append digital signatures to SOAP messages. Specifically, SOAP-DSIG defines a data format for appending an XML signature to a SOAP message. Because a digital signature is based on public key cryptography, it generally takes much more time to compute a digital signature than a MAC. However, because the sender and recipient no longer need to share a secret key, you can identify the creator of a message, that is, it is guaranteed that the signer is the creator.

Technologies for sender/recipient authentication
There are two widely used technologies for sender/recipient authentication. Note that an HTTP client (server) could be both a sender and a recipient.

- **Password authentication:**
  This is a widely used mechanism and is actually used on Amazon.com. Typical examples include HTTP basic authentication and form-based authentication. These can be used for sender authentication, where the HTTP client is supposed to send a message. An HTTP client can authenticate itself to an HTTP server by sending its identity and password. Because the password should be kept secret, SSL is usually used to send it.

- **SSL server/client authentication:**
  This is a technology for mutually authenticating the identities of HTTP servers and clients based on their public key certificates. In particular, SSL server authentication is widely used on the Internet, for example at Amazon.com. On the other hand, SSL client authentication is optional and is not currently used by very many Web sites. However, there are cases such as online trading where a public key certificate is distributed to each account holder and SSL client authentication is used to authenticate the identity of the account holder.
Password authentication is not directly comparable to SSL authentication in terms of security. However, system management for a system based on password authentication is easier than for a system based on SSL authentication, since SSL requires public key certificates and corresponding private keys which must be issued and managed. This tends to be demanding because a Certificate Revocation List (CRL) must also be maintained for key revocation and update.

What is non-repudiation?
In addition to the above two security requirements, non-repudiation is an important requirement in B2B applications as well. The need for non-repudiation arises because of malicious senders. Non-repudiation guarantees that no malicious sender can later disavow having created and sent a specific message. This implies that non-repudiation guarantees that the sender of a message is the same as the creator of the message.

For example, suppose that Enterprise A created and sent a purchase order to Enterprise B. When Enterprise B processes the purchase order and issues a bill for it, Enterprise A should not be able to repudiate the fact of having sent the purchase order. In order to satisfy the requirement of non-repudiation, you need both message authentication and sender authentication simultaneously. (Recipient authentication is not related to non-repudiation.)

Message authentication by using a digital signature is not sufficient for non-repudiation. Because a signature alone does not guarantee that the sender is who they say they are, the message transmission is vulnerable to techniques such as replay attacks by some malicious third party.

For example, assume that Enterprise A sends a purchase order with a digital signature to Enterprise B. Furthermore, assume that some malicious Enterprise C has somehow obtained a copy of the order. If Enterprise C re-sends the order to Enterprise B then Enterprise B will accept it as another order from Enterprise A (a replay attack by Enterprise C). Also, a malicious Enterprise A could repudiate the second order and claim that the second order is the result of a replay attack by a malicious Enterprise C, even though Enterprise A actually sent the order. Of course, message authentication by a MAC does not help with the problem of non-repudiation because, as mentioned above, nobody can determine whether the messages are created by the senders or the recipients.

Similarly, sender authentication is not sufficient for non-repudiation. Because it does not guarantee that messages were not modified en route, a malicious sender can claim that a message received by a recipient was modified en route even though the malicious sender created the message.

In summary, in order to satisfy the requirement of non-repudiation, it is necessary to simultaneously satisfy both the message authentication requirement by using a digital signature and the sender authentication requirement.

How to use SOAP-DSIG and SSL for non-repudiation
Now I'll examine the relationship between SOAP-DSIG and SSL from the perspective of non-repudiation. As a context for this examination, I'll first describe a typical scenario, in which a request/response message pair is signed using SOAP-DSIG and exchanged using HTTP. The following is an example of the request message.

Listing 1: A typical SOAP-DSIG request message

the <SOAP-ENV:Body> element contains application data, which represents an order to buy IBM stocks.
Furthermore, using SOAP-DSIG, this `<SOAP-ENV:Body>` element is signed and the generated signature (`<SOAP-SEC:Signature>` element) is included in the SOAP header. In this example, the key used to sign the message is identified by the `<ds:KeyName>` element ("Michael") and so it is guaranteed that this SOAP message was created by the user Michael. That is, SOAP-DSIG is used to satisfy the message authentication requirement. Finally, the signed SOAP message (`<SOAP-ENV:Envelope>` element) is put in the payload of an HTTP POST request and sent to an on-line trade server. Note that this HTTP request can be sent over SSL.

See Listing 1 for a typical SOAP-DSIG request message.

Listing: 1

```
POST /order HTTP/1.1
   Host: www.onlinetrade.com
   Content-Type: text/xml; charset="UTF-8"
   Content-Length: nnnn
   SOAPAction: "http://www.onlinetrade.com/order#buy"

   <SOAP-ENV:Header>
         <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:SignedInfo>
               <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
               <ds:Reference URI="#Body"/>
               <ds:Transforms/>
               <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
               <ds:Reference URI="#Body"/>
               <ds:Transforms/>
               <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
               <ds:Reference URI="#Body"/>
               <ds:Transforms/>
               <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            </ds:SignedInfo>
            <ds:SignatureValue>MC0CFFrVLtRlk=...</ds:SignatureValue>
            <ds:KeyInfo>
               <ds:KeyName>Michael</ds:KeyName>
            </ds:KeyInfo>
         </ds:Signature>
      </SOAP-SEC:Signature>
   </SOAP-ENV:Header>
      <order:buy xmlns:order="http://www.onlinetrade.com/order">
         <order:ticker-symbol>IBM</order:ticker-symbol>
         <order:quantity>100</order:quantity>
         <order:market>New York</order:market>
      </order:buy>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
On receiving this order, the online-trade server creates and sends the receipt to Michael as an HTTP response. Similarly, the receipt can be signed by using SOAP-DSIG. Listing 2 is an example of a receipt.

**Listing 2: The response to the SOAP message**

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8
Content-Length: nnnn

  <SOAP-ENV:Header>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
          <ds:Reference URI="#Body"/>
          <ds:Transforms>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>KtMup4Nj6lwx3rvEPO0vbeVu8nk=</ds:DigestValue>
        </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>TY5CLLrVLtRlk=...</ds:SignatureValue>
      </ds:Signature>
    </SOAP-SEC:Signature>
  </SOAP-ENV:Header>
  SOAP-SEC:id="Body">
    <receipt:buy xmlns:receipt="http://www.onlinetrade.com/receipt">
      <receipt:ticker-symbol>IBM</receipt:ticker-symbol>
      <receipt:quantity>100</receipt:quantity>
      <receipt:market>New York</receipt:market>
    </receipt:buy>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
See Listing 2 for the response to the SOAP message.

These listings show how SOAP messages are signed and exchanged over HTTP. It is important to note that you can use SOAP-DSIG and SSL simultaneously by exchanging the above HTTP messages over SSL. Table 1 summarizes which security requirements can be satisfied by SOAP-DSIG and SSL. SSL provides confidentiality and sender/recipient authentication. SSL also has functionality for adding MACs to transmitted messages. On the other hand, SOAP-DSIG can be used to add not only MACs but also digital signatures to transmitted messages, but it is not sufficient for sender/recipient authentication because it is vulnerable to attacks such as replay attacks. Therefore, SOAP-DSIG and SSL complement the functionalities that the other system lacks.

<table>
<thead>
<tr>
<th>Technology</th>
<th>Satisfied security requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL</td>
<td>Confidentiality, sender/recipient authentication, and message authentication by MAC</td>
</tr>
<tr>
<td>SOAP-DSIG</td>
<td>Message authentication by digital signature and MAC</td>
</tr>
</tbody>
</table>

Remember that in order to satisfy the requirement of non-repudiation, at a minimum you need to simultaneously guarantee both message authentication by using a digital signature and sender authentication. Therefore, using SOAP-DSIG and SSL (with client authentication) simultaneously is the first step towards realizing non-repudiation. Specifically, you use SOAP-DSIG for message authentication by using a digital signature and SSL client/server authentication for sender/recipient authentication. Note that neither SOAP-DSIG nor SSL can insure non-repudiation by itself.

In addition, recall that it is important to guarantee that the signer of a SOAP message is always the same as the sender of the message. For this purpose, I recommend using a common private key and the corresponding public key certificate for SOAP-DSIG and SSL. Specifically, in the above example, the private key used to sign the order in the HTTP request should be the same as the one for the SSL client authentication. Similarly, the private key used to sign the receipt in the HTTP response should be the same as the one for the SSL server authentication. From the perspective of the signature validation, in order to validate the signature of the order (or signature of the receipt), the validating party may use the public key certificate of the SSL client (or the public key certificate of SSL server) as authenticated via SSL. In this case, the <ds:KeyInfo> elements in the above example messages can be omitted.

Towards realizing more secure B2B applications
We need to ask the question whether or not to using SOAP-DSIG and SSL simultaneously is sufficient for non-repudiation in B2B applications. Unfortunately, the answer is "no" for security in a strict sense. I'll consider now attacks by malicious recipients and describe how to protect applications from such attacks. Application designers and developers must be responsible for this protection because SOAP-DSIG does not define anything to cover such attacks.

As mentioned above, a replay attack by some malicious third party is the most convenient attack. However, SSL can protect applications from replay attacks. Because SSL encrypts transmitted messages for confidentiality, no malicious third party can steal the messages. Even though a malicious third party could steal a message, it cannot re-send it to some party unless it can break the SSL client/server authentication. Therefore, it seems that using
SOAP-DSIG and SSL simultaneously is sufficient for non-repudiation, so I will now present two attacks by malicious recipients (not by third parties).

Consider a malicious recipient who claims that it has received a message twice from a sender even though the sender has actually sent it only once. Because the digital signature scheme guarantees nothing about how many times the message was signed and sent by the sender, nobody can determine whether the malicious recipient's claim is true or not. As a result, the malicious recipient may succeed in making the above claim. Alternately, a malicious sender may be able to claim that it has sent a message only once even though it has actually sent the message more than one time. In order to protect applications from this kind of attack or ambiguity, application designers or developers should add a nonce to application data that is to be signed. A nonce is a non-repeating string freshly generated by the sender (the signer) such that the intended recipient can check its uniqueness. Typically, the nonce can be implemented as a counter (a sequence number) or as a timestamp. By adding a nonce, you can distinguish between the same messages sent more than one time.

Again, consider a malicious recipient who receives a signed SOAP message and forwards it to another malicious party. What happens if the malicious party claims that it has received the signed message from the sender? Because the digital signature scheme guarantees nothing about who is the intended recipient of the signed message, nobody can determine whether the malicious party's claim is true or not. As a result, the malicious party may succeed in making the above claim. In order to protect applications from this kind of attack or ambiguity, application designers or developers should add the identity of the intended recipient to application data to be signed. The identity may be represented as the name of the recipient, the public key certificate of the recipient, or in some other way.

As described above, it is important for security against repudiation to add both a nonce and the identity of the intended recipient to application data to be signed.

Listing 3 is an extended example of the above order message. Note that a nonce (20010711-0001287634) and the identity of the recipient are added to the order information in the SOAP body. On receiving this signed order, the online-trade server needs to validate the uniqueness of the nonce and check whether its identity is specified as the intended recipient.

See Listing 3 for the order message again.

Listing 3: The order message again

```
POST /order HTTP/1.1
Host: www.onlinetrade.com
Content-Type: text/xml; charset="UTF-8"
Content-Length: nnnn
SOAPAction: "http://www.onlinetrade.com/order#buy"

                  <SOAP-ENV:Header>
                      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2000/CR-xml-
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
<ds:Reference URI="#Body">
  <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  <ds:DigestValue>j6lwx3rvEP00vKtMup4NbeVu8nk="</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>MC0CFFrVLtRlk=...</ds:SignatureValue>
</ds:Signature>
</SOAP-SEC:Signature>
</SOAP-ENV:Header>
  <order:buy xmlns:order="http://www.onlinetrade.com/order">
    <order:ticker-symbol>IBM</order:ticker-symbol>
    <order:quantity>100</order:quantity>
    <order:market>New York</order:market>
    <order:nonce>20010711-0001287634</order:nonce>
    <order:recipient>http://www.onlinetrade.com/</order:recipient>
  </order:buy>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>