Security and Reliability for Web Services

● Takayuki Maeda ● Yoshihide Nomura ● Hirotaka Hara

(Manuscript received June 22, 2003)

Web services are expected to become an important information technology (IT) in the future. There are, however, several security problems when Web services are used on the Internet, for example, interception, disguise, and repudiation. In this paper, we describe the problems that cannot be resolved by traditional technology such as SSL (Secure Socket Layer)/TLS (Transport Layer Security) and describe latest technologies such as XML digital signatures and XML encryption that can overcome these problems. We also show how these technologies are being used with Fujitsu's Interstage Application Server, which provides a CORBA (Common Object Request Broker Architecture) and J2EE (Java 2 Enterprise Edition) compliant infrastructure that delivers the highest level of performance, robustness, and scalability. Moreover, we describe our activities for establishing two standards for these new technologies: Web Services Security and Web Services Reliability.

1. Introduction

The rapidly growing broadband Internet services enable businesses to conduct large-scale message exchanges for electronic commerce. Web services are very attractive for Internet message-exchange systems in various business areas. Because Web services are exposed over the Internet and individual Web services are loosely coupled via SOAP (Simple Object Access Protocol)[1)] communications, it is easy to integrate multiple Web services and create a new business process.

However, various problems arise when business partners conduct business over the Internet using Web services. One of these problems is interception and alteration on routers, proxies, and gateways that exist between a sender and a receiver (**Figure 1**). Two other problems are that because HTTP (HyperText Transfer Protocol) is a stateless protocol that does not guarantee message delivery, SOAP messages can disappear or be duplicated before they reach the receiver. These problems can lead to the problem of "repudiation," in which a sender has sent a message but denies sending it or a receiver has received a message but denies receiving it (**Figure 2**).

In this paper, we explain Fujitsu's Web Services Security solutions, which resolve these problems and propose the Web Services Reliability, which is a SOAP-based protocol for exchanging SOAP messages with guaranteed delivery, no duplication, and guaranteed message ordering. We also describe an implementation of these technologies on Fujitsu's Interstage Application Server, which provides a CORBA (Common Object Request Broker Architecture) and J2EE (Java 2 Enterprise Edition) compliant infrastructure that delivers the highest level of performance, robustness, and scalability.

In Section 2, we describe some of the problems that users encounter when they communicate using SOAP over the Internet. In

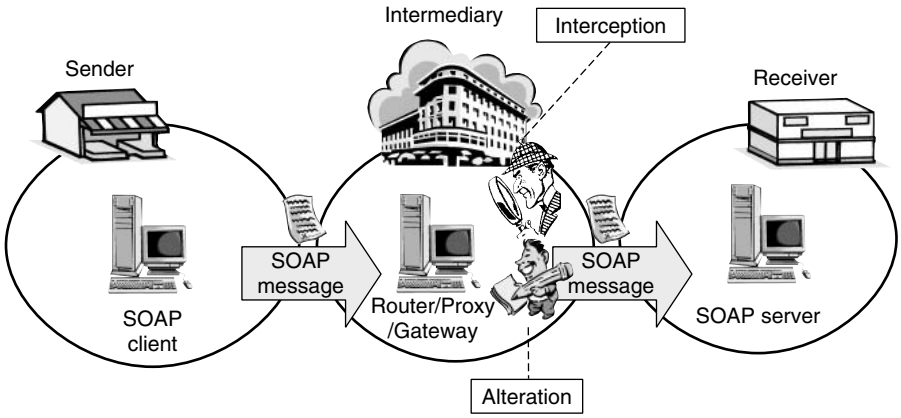

Figure 1
Interception and alteration security problems with SOAP.

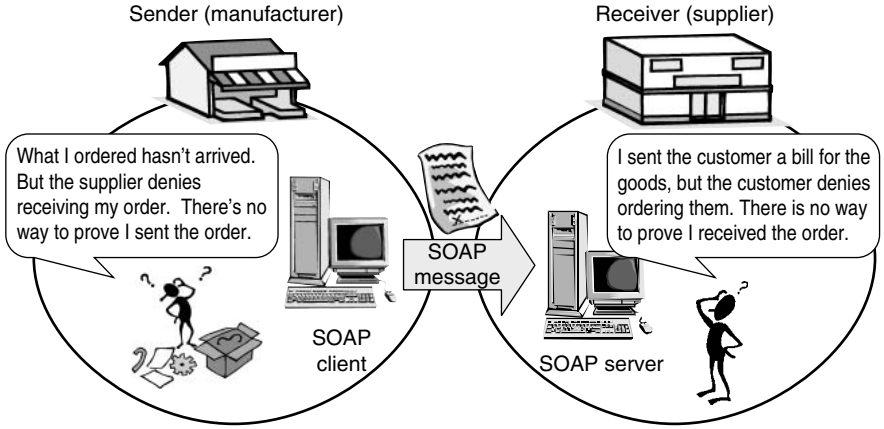

Figure 2
Repudiation security problems with SOAP.

Section 3, we describe the Web Services Security and Web Services Reliability technologies, which resolve the security problems described in Section 2. In Section 4, we describe some security solutions that implement the Web Services Security and Web Services Reliability on Interstage Application Server. In Section 5, we introduce some standardization activities in the OASIS (Organization for the Advancement of Structured Information Standards). In Section 6, we give our conclusions and describe our plans for future work.

2. Security problems when using SOAP on the Internet

SOAP is a lightweight protocol for exchanging information in a decentralized, distributed enviroment. It is an XML (Extensible Markup Language)-based protocol that has become the defacto standard for interaction between different systems over the Internet. If two parties want to keep SOAP messages confidential while communicating over the Internet, they can use SSL (Secure Socket Layer) [2]/TLS (Transport Layer Security)[3] to prevent malicious interception and alteration of the data to be communicated. SSL/TLS provides a secure channel over which the sender's data is encrypted and decrypted by the receiver. This is an adequate solution when there are only two parties in a communication.

However, SSL/TLS is not always adequate for Web services. For example, the SOAP specifications can define a SOAP intermediary between

the sender and the receiver and such an intermediary is vulnerable to man-in-the-middle attacks. Additionally, SOAP over HTTP and SOAP over SSL/TLS do not provide a message delivery guarantee function or a non-repudiation function. In the following sections, we describe why SSL/TLS does not provide Web services a suitable method of security, delivery guarantee, or non-repudiation functionality.

2.1 Security problems when a SOAP intermediary exists

When two enterprises conduct electronic commerce using Web services, there may be intermediaries between them that, for example, attach trusted time information or information authenticating that the sender has made a payment. In such a business model, even if an encryption technology such as SSL/TLS is used on the communication line, it is not possible to prevent interception and alteration by the intermediary.

In SSL/TLS, the certification authority issues certificates that certify a sender and a receiver. Then, the sender and receiver exchange their certificates before starting communications. Because the certificates certify that the public keys used by the sender and receiver belong, respectively, to the sender and receiver, the authentication and encryption performed using the corresponding private keys guarantees that these private keys belong to the sender and receiver.

If SSL/TLS authentication is used, the sender and receiver can be authenticated in the Transport Layer. This can prevent other parties from disguising the sender and receiver and also prevent interception and Alteration over the Transport Layer. However, if there is an intermediary, the authentication is performed between the sender and intermediary and between the intermediary and receiver; therefore, direct authentication between the sender and receiver is impossible. Moreover, the intermediary needs to decrypt the data sent by the sender, so it is impossible for the sender and receiver to prevent interception and Alteration by the intermediary by using SSL/TLS.

2.2 Disappearance and duplication of SOAP messages

Web services usually use SOAP over HTTP. However, they do not provide a message delivery guarantee function by themselves. Therefore, a SOAP message and the receiver's responses to the message may disappear over the communication path. Additionally, if a response message disappears, the sender may send the same SOAP message again and the receiver may process it twice. Moreover, multiple messages that the sender regarded as lost may reach the receiver multiple times.

These problems can become critical, for example, when Web services are used in B2B (business-to-business) applications. At present, when developers are working on an application that requires a message delivery guarantee function, they have to implement it by themselves.

2.3 Repudiation

Repudiation occurs when a sender denies sending a message or a receiver denies receiving a message. Repudiation is dangerous because it can lead to a contract being rescinded unilaterally and intentional duplication of orders. Fortunately, repudiation can be prevented by saving the communication data in a verifiable manner.

To prevent repudiation by using SSL/TLS, the communication data and handshakes in the Transport Layer must be saved. However, because communication data is encrypted using a shared key agreed on by handshakes, strictly speaking, it is impossible to verify whether the sender or receiver created the saved communication data.

3. Solutions to SOAP security problems on the Internet

This section introduces Web Services Secu-

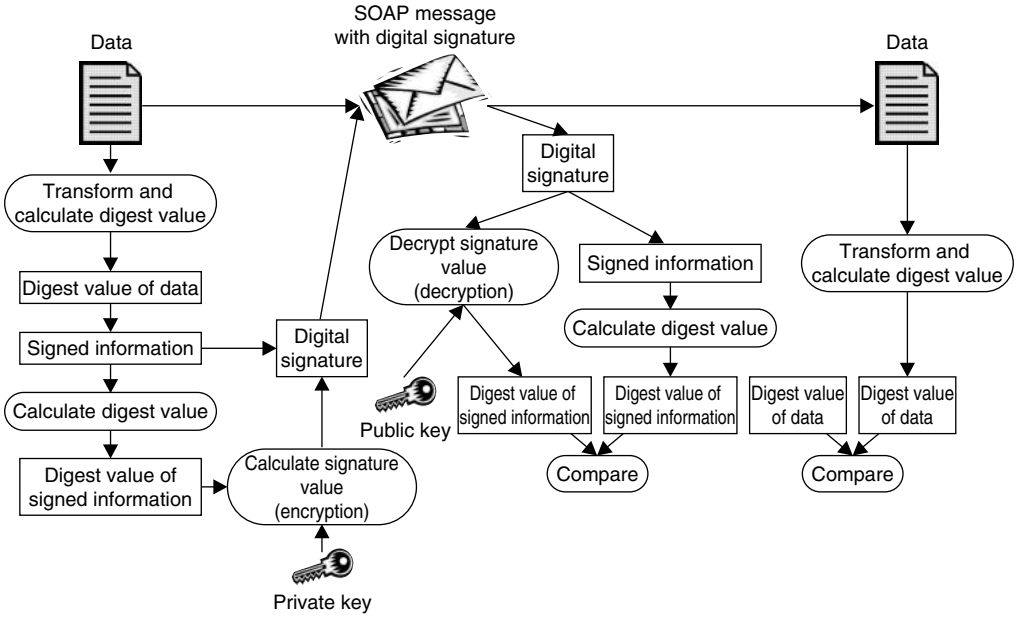

Figure 3
Mechanism of XML digital signatures.

```
(01) < SOAP-ENV:Envelope xmlns:SOAP-ENV="http://..."
                                          xmlns:wsu="http://...">
(02)  < SOAP-ENV:Header>
(03)   <wsse:Security xmlns:wsse="http://...">
(04)    <wsse:BinarySecurityToken
                              wsu:Id="cert" Type="wsse:X509v3"
                              EncodingType="wsse:Base64Binary>>
(05)     MIIEZzCCA9CgAWIBAg.....
(06)    </wsse:BinarySecurityToken>
(07)    <ds:Signature xmlns:ds="http://...">
(08)     <ds:SignedInfo>
(09)      <ds:CanonicalizationMethod
                               Algorithm="http://.../xml-exc-c14n#""/>
(10)      <ds:SignatureMethod Algorithm="http://..."/>
(11)      <ds:Reference URI="#body">
(12)       <ds:Transforms>
(13)        <ds:Transform Algorithm="http://.../xml-exc-c14n#"/>
(14)       </ds:Transforms>
(15)       <ds:DigestMethod Algorithm="http://..."/>
(16)       <ds:DigestValue>LyLsF0Pi4wPU...</ds:DigestValue>
(17)      </ds:Reference>
(18)     </ds:SignedInfo>
(19)     <ds:SignatureValue>DJbchm5gK...</ds:SignatureValue>
(20)     <ds:KeyInfo>
(21)      <wsse:SecurityTokenReference>
(22)       <wsse:Reference URI="#cert"/>
(23)      </wsse:SecurityTokenReference>
(24)     </ds:KeyInfo>
(25)    </ds:Signature>
(26)   </wsse:Security>
(27)  </ SOAP-ENV:Header>
(28)  <SOAP-ENV:Body wsu:Id="body" xmlns:wsu="http://...">
(29)   ...
(30)  </SOAP-ENV:Body>
(31) </ SOAP-ENV:Envelope>
```

Figure 4
Example of SOAP message with XML digital signature.

rity, which is a solution to the problems described in the previous section that is being discussed in OASIS. Then, we propose Fujitsu's Web Services Reliability, which ensures that a SOAP message is delivered to the intended receiver without being duplicated. Web Services Reliability also ensures that when a sender sends a message and the intended receiver receives it, the sender cannot repudiate sending it and the receiver cannot repudiate receiving it.

3.1 XML digital signatures

XML digital signatures[4] ensure the integrity of an XML document; that is, they ensure that a signed XML document has been approved by the signer and has not been altered since it was signed. The Web Services Security specification defines a method that associates an XML digital signature with a SOAP message.

Figure 3 shows the mechanism of an XML digital signature, and **Figure 4** shows a SOAP message to which an XML digital signature has been attached.

The steps for generating an XML digital signature are as follows:

1) For each data object, apply transformations and calculate the digest value of the resulting value. These results are described in a <Reference> element. In the example shown in Figure 4, the SOAP Body element referenced by the <Reference> element in line (11) is digested after canonicalization.
2) Create the signed information, which consists of the algorithm identifiers for the canonicalization and the calculation of the signature and the <Reference> element(s). The signed information in Figure 4 is the <SignedInfo> element between lines (08) and (18).
3) Canonicalize and calculate the digest value of the signed information, and then calculate the signature value by using the sender's private key.
4) Construct the digital signature, which includes the signed information, the information about the sender's private key, and the signature value. The digital signature in Figure 4 is the <Signature> element.
5) Attach the <Signature> to the <Security> header defined in OASIS Web Services Security. This header can contain security information such as security tokens, a digital signature, and/or the details of an encryption algorithm for the receiver. In this example, the <Security> header has a <BinarySecurityToken> element that includes the signer's X.509 certificate and is referenced by <KeyInfo> in the <Signature> element.

The steps for verifying the XML digital signature are as follows:

1) Extract the digital signature from the SOAP message.
2) Validate the <Reference> element(s) in the signed information using the following steps:
   Step1: For each <Reference> element, obtain the data object, apply transformations, and then calculate the digest value of the resulting value.
   Step2: Compare the generated digest value with the digest value in <Reference>. If there is any mismatch, validation fails.
3) Canonicalize and calculate the digest value of the signed information.
4) Compare the generated digest value with the digest value obtained by decrypting the signature values with the public key included in the sender's X.509 certificate.

If the signed SOAP message has been altered, the difference between the digest value in the signature and the value calculated by the receiver is detected and the receiver recognizes that the SOAP message has been altered.

Additionally, when the signer signs the SOAP message, the signer uses information known only to the signer (e.g., the X.509 certificate), or information known only to the signer and the verifier who verifies the XML digital signature (e.g., the username and password). Without this information, an intermediary positioned between the sender and the receiver of the signed SOAP message will not be able to alter the digested value and pretend to be the sender.

3.2 XML encryption

XML encryption[5] ensures the confidentiality of an XML document. The Web Services Security specification defines a method that associates XML encryption elements with a SOAP message.

Figure 5 shows the mechanism of the XML encryption, and **Figure 6** shows an example of an encrypted SOAP message.

The steps for encrypting data are as follows:

1) Generate a secret key, if one has not already been exchanged between the sender and receiver.
2) Encrypt the data using the secret key.
3) Construct an <EncryptedData> element that includes the encryption algorithm and the encrypted data, and then replace the original data with it. In the example shown in Figure 6, the contents of the SOAP Body have

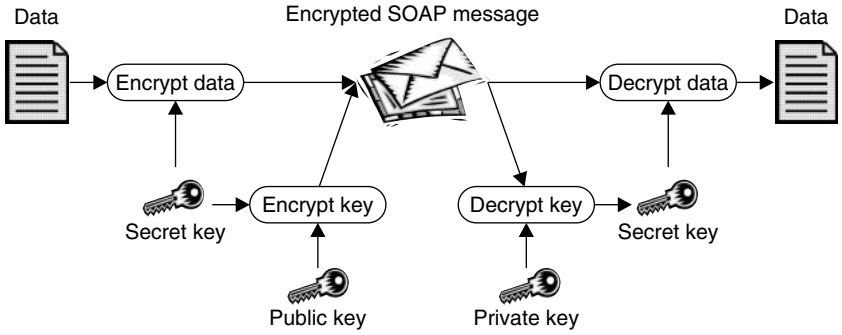

Figure 5
Mechanism of XML encryption.

```
(01) <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://...">
(02)  <SOAP-ENV:Header>
(03)   <wsse:Security xmlns:wsse="http://...">
(04)    <enc:EncryptedKey xmlns:enc="http://...">
(05)     <enc:EncryptionMethod Algorithm="http://..."/>
(06)     <ds:KeyInfo xmlns:ds="http://...">
(07)      <wsse:SecurityTokenReference>
(08)       <wsse:KeyIdentifier EncodingType="wsse:Base64Binary"
                              ValueType="wsse:X509v3">
(09)        MIGfMa0GCSq...
(10)       </wsse:KeyIdentifier>
(11)      </wsse:SecurityTokenReference>
(12)     </ds:KeyInfo>
(13)     <enc:CipherData>
(14)      <enc:CipherValue>...</enc:CipherValue>
(15)     </enc:CIpherData>
(16)     <enc:ReferenceList>
(17)      <enc:DataReference URI="#body"/>
(18)     </enc:ReferenceList>
(19)    </enc:EncryptedKey>
(20)   </wsse:Security>
(21)  </SOAP-ENV:Header>
(22)  <SOAP-ENV:Body>
(23)   <enc:EncryptedData Id="body"
                          xmlns:enc="http://..." Type="http://...">
(24)    <enc:EncryptionMethod Algorithm="http://...">
(25)    <enc:CipherData>
(26)     <enc:CipherValue>...</enc:CipherValue>
(27)    </enc:CIpherData>
(28)   </enc:EncryptedData>
(29)  </SOAP-ENV:Body>
(30) </SOAP-ENV:Envelope>
```

Figure 6
Example of encrypted SOAP message.

been replaced with the <EncryptedData> element.

4) If required, encrypt the secret key by using the receiver's public key.
5) Construct an <EncryptedKey> element that includes the encrypted secret key, and attach it to the <Security> header.

The steps for decrypting the data are as follows:

1) Decrypt the secret key in the <EncryptedKey> element by using the receiver's private key, if this element exists.
2) Decrypt the encrypted data in the <EncryptedData> element by using the secret key.
3) Replace the <EncryptedData> element with the decrypted data

Only the possessor of the private key corresponding to the certificate used by the encryptor can decrypt this encrypted data. Even if there are intermediaries between the sender and the receiver, they cannot decrypt the encrypted data in the SOAP message without knowing the private key. Therefore, the sender and receiver can keep part of the SOAP message confidential. Additionally, the sender can encrypt arbitrary elements, contents of the elements, or attachment data of the SOAP message and thereby keep one part of the message confidential but reveal another part to an intermediary.

3.3 Reliable messaging function

The reliable messaging function guarantees delivery of SOAP messages without duplication. This function has a non-repudiation function that can prevent repudiation by the sender and the receiver. The function also offers a bi-directional delivery guarantee using SOAP between a client and a server. This function enables SOAP messages to be exchanged between clients and Web

servers over a firewall with guaranteed reliability.

The reliable messaging function automatically adds headers to SOAP messages (**Figure 7**).

The additional SOAP headers indicate the message type, message ID, sender ID, and receiver ID.

The message ID is a unique string that prevents duplication of messages. Also, the confirmation message sent by the receiver prevents loss of messages.

When using the non-repudiation function, a SOAP digital signature is added to the sending message and confirmation message. **Figure 8** shows an example of the non-repudiation message flow. Because each message that is sent has a digital signature, the SOAP messages stored by the sender and receiver prevent repudiation of the transmissions. We call this function a PUSH model protocol.

```
(01)  <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://..">
(02)  <SOAP-ENV:Header>
(03)  <SSRS:MessageHeader
          xmlns:SSRS="http://xml.fujitsu.com/2001/ssrs/reliable">
(04)   <SSRS:From>
(05)    <SSRS:PartyId>urn:duns:123456789</SSRS:PartyId>
(06)   </SSRS:From>
(07)   <SSRS:To>
(08)    <SSRS:PartyId>urn:duns:123456789</SSRS:PartyId>
(09)   </SSRS:To>
(10)   <SSRS:Service>order</SSRS:Service>
(11)   <SSRS:MessageData>
(12)    <SSRS:MessageId>m012345projecta@jp.fujitsu.com
                                              </SSRS:MessageId>
(13)    <SSRS:Timestamp>2001-02-15T11:12:12Z
                                              </SSRS:Timestamp>
(14)  </SSRS:MessageData>
(15)  </SSRS:MessageHeader>
```

Figure 7
Example header added to SOAP message by reliable messaging.

Our protocol also prescribes the PULL model protocol for transmitting a message to a client from a server. **Figure 9** shows a PULL model message flow example. In the PULL model, the client sends an inquiry message to the server and then the server sends a response message to the client. The client sends a confirmation message to the server as another HTTP request.

With this mechanism, even if a client does not have a Web server or is behind a firewall, the client can receive a message from a server reliably.

4. Deploying and performing security and the reliable messaging function of Web services on Interstage Application Server

The security solutions described in the previous sections are implemented on Fujitsu's Interstage Application Server, which provides the Web server function and Java execution environments that support the latest standard

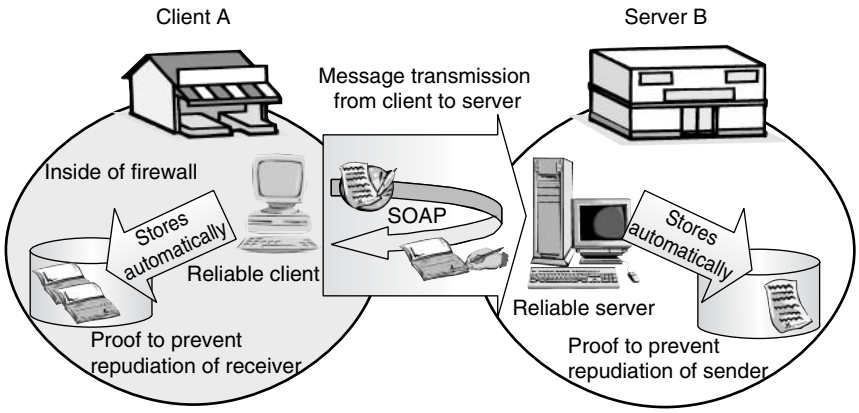

Figure 8
Message flow of reliable messaging and non-repudiation.

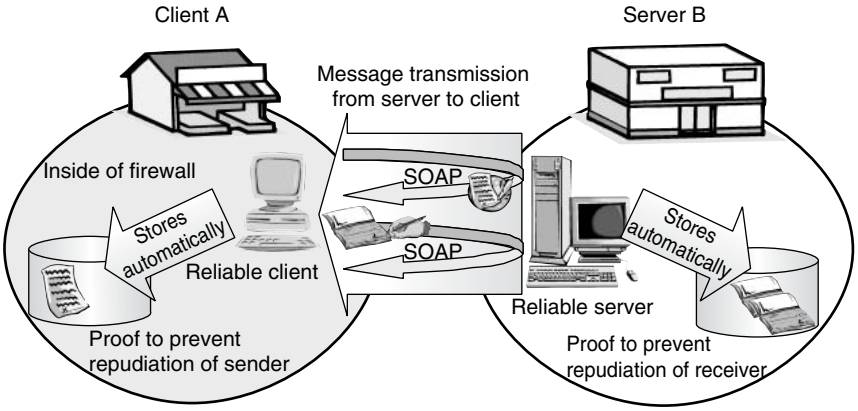

Figure 9
PULL model message flow.

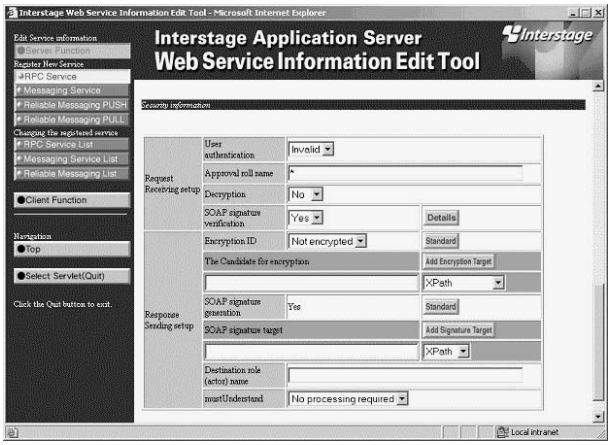

Figure 10
GUI for setting the signature and encryption environment.

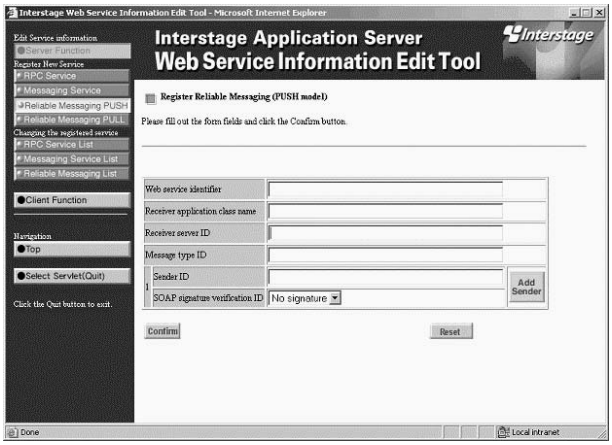

Figure 11
GUI for setting the PUSH model environment.

technologies such as SOAP, UDDI (Universal Description, Discovery, and Integration),[6] and WSDL (Web Service Definition Language).[7] In addition to these functions, our SOAP security solutions make it possible to quickly and easily make secure and reliable SOAP communications over the Internet.

On Interstage Application Server, the Web services are implemented using JAX-RPC (a Java™ API [Application Program Interface] for XML-based Remote Procedure calls) and JAXM (a Java™ API for XML messaging), which are the standard APIs for implementing Web services in a Java environment. After the Web services have been implemented, the SOAP signature and encryption environment are set using the GUI (Graphical User Interface) shown in **Figure 10**. **Figure 11** shows the GUI for the reliable messaging function. If the SOAP signature option is turned on using the GUI, a component of the security function is invoked automatically and attaches or verifies the security information at execution time. Because the implementer of the Web services does not need to change the application program to use the security function, the implementer can easily and quickly use the latest security solutions for the Web services.

The reliable messaging function for SOAP

messages can be used on Fujitsu's Interstage Application Server by using the JAXM Profile API. Therefore, by making some minor changes to a messaging application created using the JAXM API, we can turn it into a reliable messaging function.

5. SOAP security standards

Web services and Web service communications conducted using SOAP are exposed on the Internet. Since these Web services are independent of the platform, operating system, and programming language being used, it is important to standardize the various technologies associated with Web services in order to ensure application interoperability. In this section, we describe the standardization activities of Web service technologies.

5.1 The Web Services Security specification

The Web Services Security Technical Committee (WSS TC) of OASIS is currently discussing a specification called Web Services Security. Fujitsu is participating in the standardization activity of this specification as a member of this technical committee.

Web Services Security provides mechanisms that prevent various threats in the SOAP message layer. It mainly consist of the following technologies:

1) Security tokens
2) Digital signatures
3) Encryption
4) Other security considerations

A security token represents a collection of claims, which are statements that a client makes and describe other security mechanisms, for example, X.509 certificates, Kerberos tickets, usernames, and passwords.

The digital signature ensures the integrity of a SOAP message so it can be verified that it was sent by the possessor of the security token, and the encryption keeps the SOAP message confidential. Additionally, Web Services Security provides mechanisms that prevent replay attacks using timestamps or nonces.

5.2 SOAP reliable messaging

Fujitsu, Hitachi, NEC, Oracle, Sonic Software, and Sun Microsystems have jointly proposed the reliable messaging protocol described in this paper to OASIS to promote WS-Reliability (Web Services Reliability).[8)]

IBM and other companies have recently published another reliable messaging protocol called WS-ReliableMessaging.[9)] However, it has not been standardized yet.

These two specifications are expected to be unified into a single specification in the near future.

6. Conclusions

Web services are expected to become an important information technology. There are, however, several security problems when Web services are used on the Internet, for example, disguises and repudiation.

In this paper, we described some advanced technologies for solving these problems, for example, XML digital signatures and XML encryption. We also showed how Fujitsu's Interstage Application Server employs these technologies. In the future, we expect that the security provided by these technologies will encourage businesses to make extensive use of Web services.

It is important to establish standards for these new technologies. Fujitsu has been contributing to the activities of OASIS, for example, we have proposed the Web Services Reliability specification, and we will continue to make such contributions.

References

1) Simple Object Access Protocol (SOAP) 1.1, W3C Notes, May 2002.
http://www.w3.org/TR/SOAP/
2) The SSL Protocol Version 3.0, November

1996.
http://wp.netscape.com/eng/ssl3/draft302.txt

3) The TLS Protocol Version 1.0, January 1999, IETF standard.
http://www.ietf.org/rfc/rfc2246.txt
4) XML-Signature Syntax and Processing, W3C Recommendations, February 2002.
http://www.w3.org/TR/xmldsig-core/
5) XML Encryption Syntax and Processing, W3C Recommendations, December 2002.
http://www.w3.org/TR/xmlenc-core/
6) UDDI Version 2 Specifications / UDDI Version 3 Specification OASIS standards.
7) Web Services Description Language (WSDL) Version 1.2, W3C Working Draft, July 2002.
http://www.w3.org/TR/wsdl12/
8) Colleen Evans, Dave Chappell, Doug Bunting, George Tharakan, Hisashi Shimamura, Jacques Durand, Jeff Mischkinsky, Katsutoshi Nihei, Kazunori Iwasa, Martin Chapman, Masayoshi Shimamura, Nicholas Kassem, Nobuyuki Yamamoto, Sunil Kunisetty, Tetsuya Hashimoto, Tom Rutt, and Yoshihide Nomura: Web Services Reliability (WS-Reliability) Ver1.0, January 2003.
http://xml.fujitsu.com/en/about/WS-ReliabilityV1.0.pdf
9) Specification: Web Services Reliable Messaging Protocol (WS-ReliableMessaging), March 2003.
http://www-106.ibm.com/developerworks/webservices/library/ws-rm/

Takayuki Maeda received the B.E. degree in Mechanical Engineering and the M.S. degree in Information Sciences from Tohoku University, Sendai, Japan in 1996 and 1998, respectively. He joined Fujitsu Ltd., Kawasaki, Japan in 1998, where he was engaged in development of Global Server until 2001. Since then, he has been engaged in research and development of middleware for Web services.

Yoshihide Nomura received the B.E. and M.E. degrees in Engineering from Aoyama Gakuin University, Tokyo, Japan in 1996 and 1998, respectively. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1998, where he has been engaged in research and development of Web application frameworks and middleware. He is a member of the Information Processing Society of Japan (IPSJ).

Hirotaka Hara received the B.S. degree in Information Science in 1984 and the Ph.D. in Information Engineering in 1992 from the University of Tokyo, Tokyo, Japan. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1984, where he has been engaged in research and development of artificial intelligence and distributed enterprise systems. He is a member of the IPSJ. He received the IPSJ Convention Award and the JSAI Annual Conference Award in 1993.