Application of Web Services Security: A Case of Travel Industry

Kojiro Nakayama and Takeshi Ishizaki

Systems Development Laboratory, Hitachi, Ltd., Japan

{kojiro, ishizaki}@sdl.hitachi.co.jp

Michiko Oba

Software Division, Hitachi, Ltd., Japan

michiko.oba.cq@hitachi.com

Abstract: Web Services Security (WS-Security) is a specification that protects SOAP messages to ensure end-to-end security for web services. WS-Security was approved as the OASIS standard in April 2004 and the first stage of standardization has been completed. Although the interoperability of WS-Security itself has been examined, business applications of WS-Security have not yet been fully investigated. Applying WS-Security to actual businesses is the next step. We conducted a large-scale demonstration experiment with web services using a travel industry model. We applied WS-Security to travel booking transactions and succeeded in ensuring end-to-end security by signing and encrypting credit card numbers. We give an overview of the experiment, point out the problems experienced and provide a possible solution. The experiment revealed that problems still remain with respect to communication via an intermediary.

Keywords: web services, security, WS-Security

Kojiro Nakayama has been a researcher of Systems Development Laboratory, Hitachi, Ltd. since 2002. He received B. Sc. and M. Sc. degrees in Dept. of Physics, from Waseda University in 2000 and 2002 respectively. His research interests include the web services security and web application security.

Takeshi Ishizaki has been a researcher of Systems Development Laboratory, Hitachi, Ltd. since 1989. He received M. Sc. in Engineering from Kyoto University in 1989. His research interests include IT platform management technology. Member of IPSJ, IEICE and ACM.

Michiko Oba has been an engineer of Software Division, Hitachi, Ltd. since 1982. She received B. Home ec. in Dept. of Sciences for Home Economics from the Japan Women's University in 1982. She received Dr. Eng. from Osaka University in 2001. Her research interests include workflow systems and services oriented architecture.

1. Introduction

Web services are XML [14] based system integration technology which utilizes SOAP [15] message, a standardized communication protocol, to achieve heterogeneous system integration. Web services have become popular in recent years. A number of web services systems have already come into practical use. Ensuring security is a critical issue when web services are used in real business systems.

Web Services Security (WS-Security) [8] is a specification that protects SOAP messages, ensuring end-to-end security for web services. It defines the ways XML Signature [18] and XML Encryption [17] are used within SOAP messages. It also defines the ways security tokens (e.g. Usernames/Password and X.509 certificate) are attached to SOAP messages [10][11].

WS-Security was originally developed by IBM, Microsoft, and VeriSign [1][2]. These companies submitted WS-Security to OASIS [7], the international standards organization for e-business. OASIS Web Services Security Technical Committee (WSS TC) [6] was formed in July 2002 and they have continued developing and standardizing WS-Security. WS-Security was approved as an OASIS standard in April 2004 and the first stage of standardization has been completed. Applying WS-Security to actual business is the next step.

The interoperability of WS-Security has been a sensitive topic. OASIS WSS TC had conducted interoperability tests in parallel to develop specifications. WS-I, the organization intended to promote interoperability of web services, is developing a Basic Security Profile [19] to promote interoperability of WS-Security.

These activities have focused on interoperability between the SOAP message sender and receiver, i.e. the main concern has been whether the receiver can normally process the message sent by the sender. When we use WS-Security in actual business, we have to investigate numerous other issues and end-to-end security via an intermediary has not fully been assessed.

We conducted a large-scale demonstration experiment with web services using a travel industry model. The system consisted of three types of entities, and they exchanged the highly confidential private data via an intermediary. We applied WS-Security to travel booking transactions and succeeded in ensuring end-to-end security.

In this paper, we give an overview of the experiment, point out the problems we experienced, and provide a possible solution. The experiment revealed that problems still remain with respect to communication via an intermediary.

2. Business Model for Travel Industry

Electronic Data Interchange (EDI) has been used in the travel industry since early on. A standard data format has been required to eliminate the data formats unique to each system. In Europe and the United States, OpenTravel Alliance (OTA) [12] has been developing XML based specifications for individual travel booking. However, in Japan, the package tour travel is more popular than individual excursions. Therefore, the OTA specification is not suitable for the Japanese travel industry. TravelXML [23] was developed by the Japan Association of Travel Agents (JATA) [3] and the XML Consortium [21], an organization founded to enlighten and spread the use of XML related technology. TravelXML defines the data formats for package tour bookings.

The Business model for the travel industry in Japan is outlined in Fig. 1. Several entities are involved in a transaction to make a package tour booking. Wholesalers purchase rooms from hotels, airplane seats from airlines and train tickets from railways. They plan package tours combining these advance purchases. Then, wholesalers commission retailers to sell the package tours. Travelers make travel arrangements through retailers.

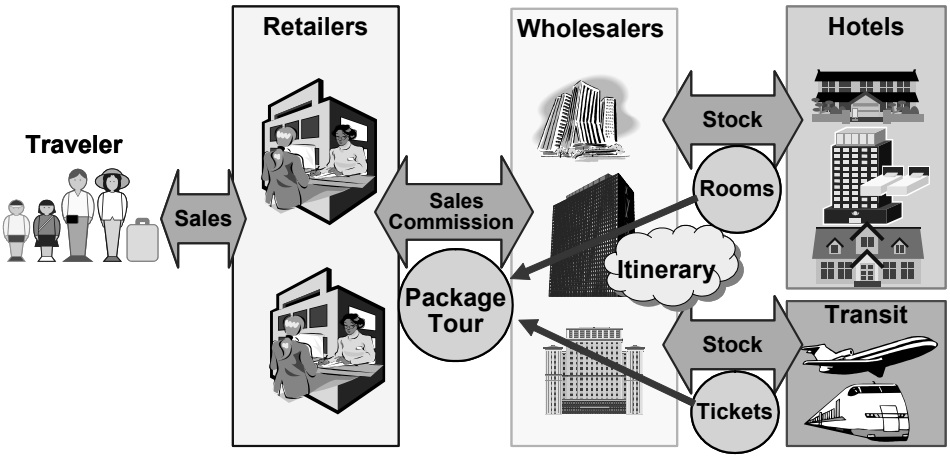

Figure 1: Business Model for Travel Industry [22]

3. Details of the Experiment

A large-scale experiment to demonstration web services using a TravelXML was conducted by the XML Consortium Applied Technology Group [4][5][22]. More than 30 engineers from 15 companies participated. It took 6 months from planning, design, implementation and testing. The purpose of the experiment was to verify the validity of web services and WS-Security with a realistic business model. In this paper, we discuss the security aspects of this experiment.

3.1. Demonstration Scenario

There is an overview of the demonstration system in Fig. 2. It consists of three types of entities, i.e., retailers, wholesalers and hotels. These systems exchange the data defined by TravelXML using web services.

The demonstration scenario was as follows:

1) The prospective traveler visits a retailer whose salesclerk selects a suitable package tour according to his/her requirements.

2) The retailer sends a "Booking Request" to the wholesaler.

3) The wholesaler receives the "Booking Request" and updates the number of rooms in stock.

4) The wholesaler sends a "Allotment Booking Report" to the hotel.

5) The hotel receives the "Allotment Booking Report" and updates the status of the booking.

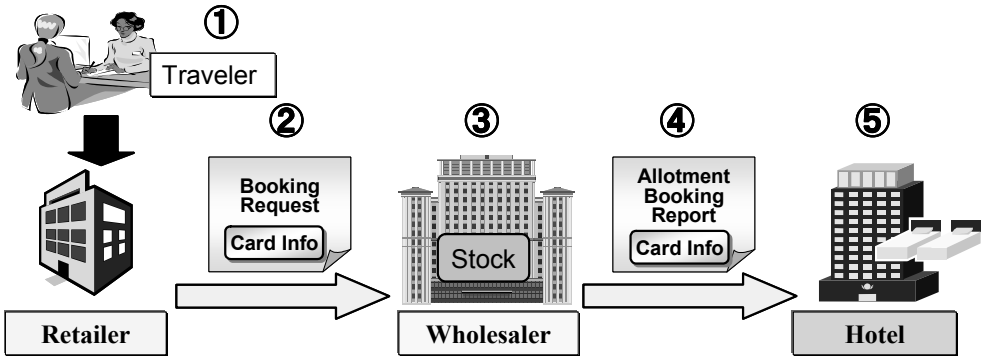

Figure 2: Demonstration Scenario [22]

In the scenario, the hotel offers an esthetic service, and prospective traveler is hoping to enjoy this service. The fare for the package tour is paid to the retailer by the prospective traveler but the fare for the esthetic service is not included in the fare for the package tour. Therefore, the prospective traveler pays for the fare for the esthetic service to the hotel directly by credit card. The credit card information is sent from the retailer to the wholesaler in the "Booking Request" and then forwarded to the hotel in the "Allotment Booking Report".

3.2. Security Scenario

Web services security can be classified into transport layer security and SOAP message layer security [20]. Because transport layer security is well established technology, we have not dealt with it here. We focused on SOAP message layer security in this experiment.

As previously described, the traveler's credit card information is sent from the retailer to the hotel. Although the credit card information is sent via the wholesaler, the wholesaler does not need to know the credit card information. We used WS-Security to protect credit card information to ensure end-to-end security. By using the XML encryption as profiled in WS-Security, credit card information is not disclosed to the wholesaler. By using the XML signature as profiled in WS-Security, the hotel is able to confirm that the credit card information has not been altered.

The security processing for each entity is as follows:

Retailer

There is an outline of the "Booking Request" SOAP message in Fig. 3. <BookingRequest> element defined by TravelXML is placed in the SOAP Body. <BookingRequest> element contains booking

information (course name, departure date, and other details.) and the traveler's credit card information.

The retailer creates a <Security> header inside the SOAP Header to place security related information. The retailer then digitally signs the <CreditCardNumber> element which is included in the credit card information, and describes signature information inside the <Security> header. Then, the retailer places the X.509 certificate as a security token inside the <Security> header for message receiver to verify signature. X.509 certificate is described as a <BinarySecurityToken> element defined by WS-Security.

The retailer then encrypts the <CreditCardNumber> element using the hotel's public key and replaces it with the <EncryptedData> element defined by XML Encryption. Then, the retailer places the encryption information inside the <Security> header.

Figure 3: Booking Request SOAP message

Wholesaler

There is an outline of security processing by the wholesaler is shown in Fig. 4. Wholesaler does process neither signature nor encryption. The wholesaler copies security related data and forwards this to the hotel.

The wholesaler receives the "Booking Request" from the retailer and constructs the <AllotmentBookingReport> element defined in TravelXML. The <AllotmentBookingReport> element includes the wholesaler's name, check in date, and other information. The wholesaler copies the credit card information included in the "Booking Request" into the "Allotment Booking Report". The wholesaler also copies the <Security> header into the "Allotment Booking Report".

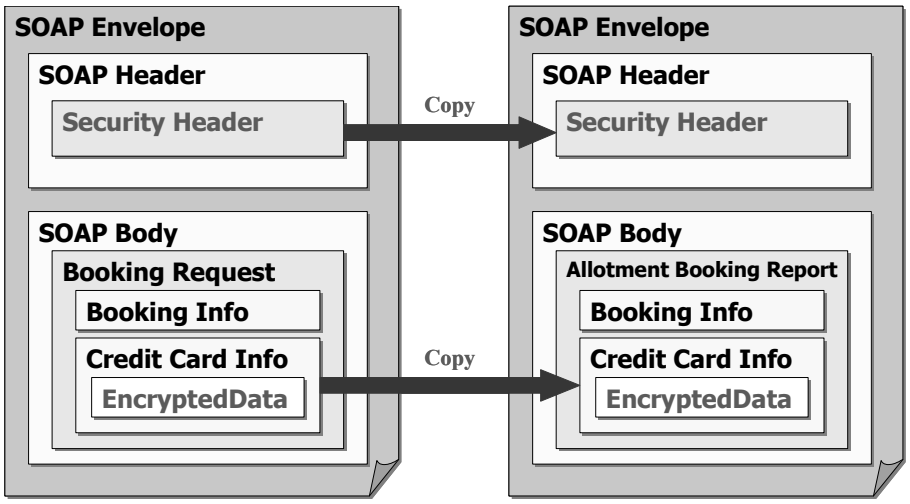

Figure 4: Processing by the Wholesaler

Hotel

The hotel receives the "Allotment Booking Report" and decrypts the credit card information. Then the hotel verifies the signature using the security token included in the <Security> header.

The XML signature and XML encryption related parameters we used in the experiment are in Table 1. These algorithms and mechanisms were selected because these are widely implemented and useful. The retailer and the hotel exchanged these parameters out-of-band.

Table 1. XML Signature/XML Encryption Parameters

Type	Process	Algorithms/Mechanisms
XML Signature	Canonicalization	Exclusive XML Canonicalization
	Signature	PKCS1 (RSA-SHA1)
	Transform	Exclusive XML Canonicalization
	Digest	SHA-1
	Key Reference	Security Token Reference with Direct Reference
XML Encryption	Key Transport	RSA Version 1.5
	Block Encryption	Triple DES
	Key Reference	Security Token Reference with KeyIdentifier

4. Results of experiment and considerations

4.1. Processing by intermediary

Copying XML element

As previously described, the wholesaler copies security related data from the "Booking Request" to the "Allotment Booking Report". During this process, many products automatically insert additional white spaces and/or change the namespace prefix inside the signed element.

There is an outline of the prefix change during the wholesaler processing in Fig. 5. In general, the data is canonicalized before being signed. However the current standard algorithm (e.g. Exclusive XML Canonicalization [13]) does not canonicalize the prefix. Prefix change has no impact on the meaning of XML data (TravelXML), but it invalidates the signature. White space insertion also invalidates the signature.

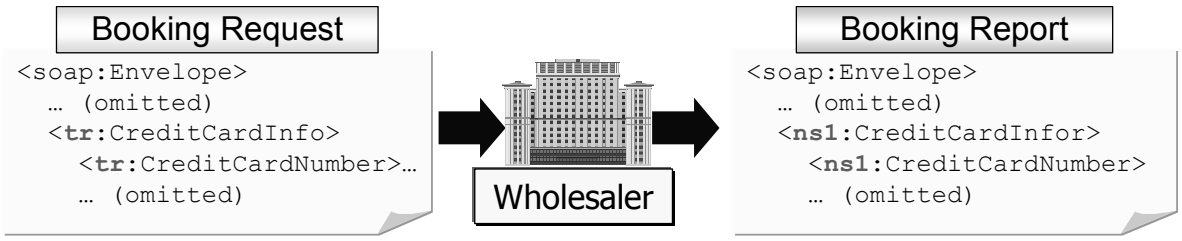

Figure 5: Prefix Change During the Wholesaler Processing

The retailer first signed then encrypted in the experiment, which had the effect of preventing the invalidation of the signature by the wholesaler. However, care must be taken when this method is used. Digest value of the plain text (credit card number) is included in the header of the message and it could be a clue to real value. To avoid this problem, it is necessary to take measures such as encrypting the digest value included in the header.

Schema validation

The Retailer encrypts the credit card number element and replaces it with the <EncryptedData> element. <EncryptedData> is an element, which is defined by the XML encryption and not defined in TravelXML. Because the encrypted message is not valid for TravelXML schema definition, the wholesaler cannot perform the schema validation. To perform the schema validation for the encrypted message by the wholesaler, it is necessary to rewrite the TravelXML schema definition so that the <EncryptedData> element can be dealt with.

4.2 Interoperability of WS-Security

Although several products support WS-Security already, each product implements a different version of WS-Security. There are numerous versions of WS-Security specifications if draft versions are considered. Interoperability may not be achieved if the versions differ.

In the experiment it was the following three factors that defeated the interoperability. (Note that there are other factors which defeat the interoperability outside the scope of this experiment.)

Namespace URI

Each WS-Security specification uses different namespace URI.

QName vs. URI

WS-Security defines a number of type attribute, e.g. ValueType attribute to indicate the type of security token, and EncodingType attribute to indicate the encoding format of the binary data.

In the OASIS WS-Security Working Draft 17 [9] or before, QNames are used for the value of these type attributes, e.g. “wsse:X509v3” to indicate X.509 v3 certificate, and “wsse:Base64Binary” to indicate XML Schema base 64 encoding. Assume the namespace prefix “wsse” binds to the URI that each WS-Security specification uses.

In the OASIS standard WS-Security, URIs are used instead of QNames, e.g. “...#X509v3” and “...#Base64Binary”. Assume “...” means the URI that each WS-Security specification uses.

ValueType attribute of the KeyIdentifier

In the OASIS WS-Security Working Draft 17 or before, ValueType attribute of the KeyIdentifier indicates the type of security token being referenced. In the OASIS standard WS-Security, it indicates the type of Key Identifier itself.

Because WS-Security was standardized by OASIS, the WS-Security version may be unified into OAISIS standard WS-Security, solving this problem.

4.3. Key Management

The retailer encrypts the <CreditCardNumber> element using hotel's public key. In the experiment, we suppose that retailer already have the hotel's public key. However it is difficult for each retailer to manage the key appropriately. Therefore, it is better to separate key management system from each application system. Key management is outside the scope of WS-Security. XML Key Management Service (XKMS) [16] is the candidate technology to solve this problem.

5. Conclusion

We conducted experiments to demonstration a web services system using TravelXML. In the experiment, we used WS-Security to ensure end-to-end security. We clarify the problems with the application of WS-Security and suggested the possible solutions to these.

Interoperability of WS-Security did not pose a serious problem. Although there were unresolved issues with the differences in WS-Security versions, these are expected to be solved soon. There were problems with processing by the wholesaler for the signed data. The intermediary needs to be careful not to invalidate the signature. Signing first and encrypting later can effectively retain the validity of the signature. Another problem with intermediary processing is that they cannot validate the schema for the received using unchanged schema. We need to rewrite the schema to deal with encrypted data.

The solution shown here is not a definitive and problems remain with respect to communication via an intermediary.

Acknowledgements

This demonstration experiment was conducted by the XML Consortium, Applied Technology Group. Fifteen companies have participated in this experiment. They were: Ad-Sol Nissin Corporation, Brainyworks Ltd., Hitachi Systems & Services Ltd., Hitachi Ltd., IBM Japan, Ltd., Infoteria Corporation, IONA Technologies Japan, Ltd., NEC Corporation, NET TIME Corp., Nihon Unisys Software Ltd., Nihon Unisys Ltd., Oracle Corporation Japan, PFU Active Labs Limited, Tokyo Electron Limited, and Toshiba Solutions Corporation. The authors would like to thank all the contributors to this experiment.

REFERENCES

1. IBM, Microsoft, VeriSign, Web Services Security (WS-Security), http://msdn.microsoft.com/ws/2002/04/Security/

2. IBM, Microsoft, VeriSign, Web Services Security Addendum, http://msdn.microsoft.com/ws/2002/07/Security/

3. Japan Association of Travel Agents (JATA), http://www.jata-net.or.jp/english/

4. KOJIRO NAKAYAMA et al : **Application of Web Services Security Using Travel Industry Model**, SAINT 2005 Workshop, pp 358-361 (2005).

5. MICHIKO OBA et al : **Web Services Experiment Using Standard XML: TravelXML for Travel Industry Electronic Commerce**, ENEI'05, pp.152-156 (2005).

6. OASIS Web Services Security TC, http://www.oasis-open.org/committees/wss

7. OASIS, http://www.oasis-open.org/,

8. OASIS, Web Services Security: SOAP Message Security 1.0 , http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

9. OASIS, Web Services Security: SOAP Message Security Working Draft 17, http://www.oasis-open.org/committees/download.php/3281/WSS-SOAPMessageSecurity-17-082703-merged.pdf

10. OASIS, Web Services Security: UsernameToken Profile , http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf ,

11. OASIS, Web Services Security: X.509 Token Profile , http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf

12. OpenTravel Alliance, http://www.opentravel.org/ ,

13. W3C, Exclusive XML Canonicalization Version 1.0 , http://www.w3.org/TR/xml-exc-c14n/

14. W3C, Extensible Markup Language (XML), http://www.w3.org/TR/REC-xml/ ,

15. W3C, Simple Object Access Protocol (SOAP) , http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

16. W3C, XKMS, http://www.w3.org/TR/xkms/ ,

17. W3C, XML Encryption Syntax and Processing, http://www.w3.org/TR/xmlenc-core/

18. W3C, XML-Signature Syntax and Processing, http://www.w3.org/TR/xmldsig-core/

19. WS-I, Basic Security Profile Version 1.0 Working Group Draft, http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0-2006-01-20.html

20. WS-I, Security Challenges, Threats and Countermeasures Version 1.0, http://www.ws-i.org/Profiles/BasicSecurity/SecurityChallenges-1.0.pdf

21. XML Consortium, (in Japanese), http://www.xmlconsortium.org/

22. XML Consortium, Project of Web Services Demonstration Experiment using TravelXML, Result Data (in Japanese)

23. XML Consortium, TravelXML specification (in Japanese), http://www.xmlconsortium.org/wg/TravelXML/TravelXML_index.html