社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006－DD－56（6）
2006／7／28

暗号化XMLデータのスキーマ検証方式の提案

中山 弘二郎† 荒本 道隆†† 大場 みち子†††

†（株）日立製作所 システム開発研究所 〒215-0013 神奈川県川崎市麻生区王禅寺 1099
††アドソル日進（株） 生産技術部 〒108-0075 東京都港区港南 4－1－8 リバージュ品川
†††（株）日立製作所 ソフトウェア事業部 〒140-8573 東京都品川区南大井 6-26-2 大森ベルポート A 館

E-mail: †kojiro@sdl.hitachi.co.jp, ††Aramoto.Michitaka@adniss.jp, †††michiko.oba.cq@hitachi.com

あらまし Web サービスのセキュリティを end-to-end に渡って確保するためには，XML データに対するセキュリティ処理が欠かせない．しかし XML 暗号を用いて XML データの部分暗号化を行なうと XML データのデータ構造が変わるため，暗号化後の XML データはスキーマに対する妥当性が失われる．そのため XML データを復号することのできないメッセージの中継者においてスキーマ検証が実施できないという問題があった．本研究では，暗号化に対応したスキーマを自動生成することで部分暗号化されたメッセージのスキーマ検証を行なう方法を提案する．これにより，メッセージの復号ができない中継者においてもスキーマ検証を実施することが可能になる．

キーワード XML，Web サービス，セキュリティ，XML 暗号，スキーマ検証

Proposal of Schema Validation Method for Encrypted XML Data

Kojiro NAKAYAMA† Michitaka ARAMOTO†† and Michiko OBA†††

† Systems Development Laboratory, Hitachi, Ltd. 1099 Ohzenji, Asao-ku, Kawasaki-shi, Kanagawa 215-0013 Japan
†† Productive Engineering Dept., Ad-Sol Nissin Corp. 4-1-8 Kounan Riverge, Shinagawa, Minato-ku, Tokyo 108-0075 Japan
††† Software Division, Hitachi, Ltd. Omori Bellport A Bldg. 6-26-2 Minamioi, Shinagawa-ku, Tokyo 140-8573 Japan

E-mail: †kojiro@sdl.hitachi.co.jp, ††Aramoto.Michitaka@adniss.jp, †††michiko.oba.cq@hitachi.com

Abstract XML Security is necessary to ensure the end-to-end security of web services. To provide the confidentiality against the intermediaries along the message path, XML Encryption is used to partially encrypt the XML data. However, because partially encrypted XML data is not valid for original schema definition, intermediaries cannot perform the schema validation. In this paper, we propose a new schema validation method. By using this method, intermediaries can perform the schema validation for partially encrypted XML data.

Keyword XML, Web Services, Security, XML Encryption, Schema Validation

1. はじめに

顧客ニーズの変化や企業の買収・統合など，近年の企業を取り巻く環境は激しく変化している．そのため企業システムには，これらの環境の変化に従来以上に素早く柔軟に対応することが求められている[1]．このような柔軟なシステムを実現するためのアーキテクチャとして SOA（Service Oriented Architecture）[2]が注目を集めている．SOA では，業務をサービスと呼ばれる再利用可能な単位に分割し，これらのサービスと連携することでビジネスプロセスを実現する．Web サービスは XML（Extensible Markup Language）などの標準技術を用いたシステム連携技術であり，SOA におけるサービスを実現するための基盤技術と考えられている．

Web サービスを実ビジネスで利用するにはセキュリティの確保が欠かせない．従来の Web アプリケーションでは，Web サーバーとブラウザの二者間でのセキュリティを確保すればよく，SSL（Secure Socket Layer）/TLS（Transport Layer Security）のようなトランスポートレイヤにおけるセキュリティ技術を用いてデータの完全性，秘匿性を確保することができた．しかし，Web サービスでは中継者を介したマルチホップの通信も想定されている[3]．中継者を介したマルチホップの Web サービスにおいて end-to-end に渡るデータの完全性，秘匿性を確保するためには，トランスポートレイヤにおけるセキュリティ処理では不十分であり，メッセージレイヤでセキュリティ処理を行なう必要がある[4]．WS-Security（Web Services Security）[5]は，Web サービス

におけるメッセージレイヤでのセキュリティを確保するための仕様であり，サービス間で交換される SOAP（Simple Object Access Protocol）[3]メッセージに対する暗号，署名の方法を規定している．WS-Security を用いることで，中継者を介したマルチホップの Web サービスにおいても end-to-end のセキュリティを確保することができる．

しかし，WS-Security を用いてメッセージの部分暗号化を行う場合，暗号化後の XML データはスキーマに対する妥当性が保証されない．そのため三者間 Web サービスの中継者はスキーマ検証が実施できないという問題が発生する[6][7]．本稿では，暗号化に対応したスキーマを自動生成することにより上記の問題点を解決するスキーマ検証方式の処理の詳細について述べる．

2. 背景技術

2.1. 三者間 Web サービスのセキュリティ

Web サービスでは，SOAP メッセージの最初の送信者と最終的な受信者の間に中継者が存在し，中継者が SOAP メッセージの転送を行なうケースが想定されている．このような中継者を介した三者間以上での Web サービスにおけるセキュリティを確保するには，SSL/TLS に代表されるトランスポートレイヤでのセキュリティだけでは不十分であり，WS-Security による SOAP メッセージレイヤでのセキュリティも必要になる．

Web サービスで交換されるデータの盗聴を防ぎ秘匿性を確保するには，データの暗号化が有効である．SSL/TLS を用いてトランスポートレイヤで暗号処理を行なう場合，point-to-point（直接の通信相手との間）の秘匿性を確保できる．しかし，中継者において一度データが復号されるため，end-to-end（最初の送信者から最終的な受信者までの間）の秘匿性を確保することができない．end-to-end の秘匿性を確保するには SOAP メッセージレイヤで暗号化を行なう必要がある．SOAP メッセージに対する暗号化の方法を定めた標準仕様として WS-Security がある．

WS-Security を用いた SOAP メッセージの暗号化の概要を図 1 に示す．最初の送信者は，中継者に開示したくないデータ（ここではクレジットカードの情報）を，最終的な受信者の公開鍵を用いて暗号化する．この際，SOAP メッセージ全体を暗号化するのではなく，中継者に開示したくないデータのみを XML 暗号[8]を用いて部分的に暗号化する．中継者では，クレジットカードの情報を復号することができないため，暗号化されたままの状態で最終的な受信者に転送する．最終的な受信者は，自身の秘密鍵を用いてクレジットカード情報の復号を行なう．以上の処理により，クレジットカード情報を，中継者に開示することなく最終的な受信者に送信することができる．

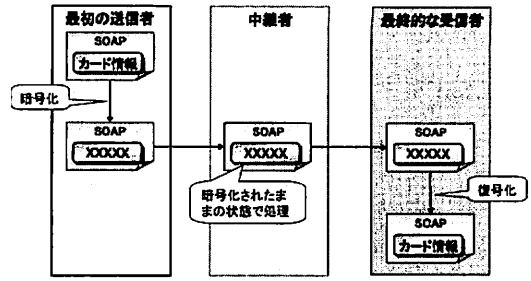

図 1　三者間 Web サービスにおける暗号処理

Fig.1　Encryption processing among three web services entites.

2.2. XML データのスキーマ検証

XML データが満たすべき制約条件はスキーマによって定義される．スキーマでは，XML データ内で使用される要素や属性の名前，データの階層構造などが定義される．XML データがスキーマで定義された制約条件を満たす場合，XML データはスキーマに対して妥当であると呼ばれる．XML のスキーマを記述する言語には，DTD[9]，XML Schema[10][11]，Relax NG[12]などがある．XML Schema によるスキーマ定義の例を図 2 に示す（説明を簡単にするためスキーマ定義の一部分のみを示している）．図 2 に示したスキーマ定義では，CreditCardInformation 要素の子要素として CreditCardAuthority 要素，CreditCardNumber 要素，ExpireDate 要素，CardHolderName 要素が出現することなどが定義されている．図 2 に示したスキーマ定義に従って生成された XML データの例を図 3 に示す．

```
<element name="CreditCardInformation">
 <complexType>
  <sequence>
   <element ref="CreditCardAuthority" minOccurs="0" />
   <element ref="CreditCardNumber" minOccurs="0" />
   <element ref="ExpireDate" minOccurs="0" />
   <element ref="CardHolderName" minOccurs="0" />
  </sequence>
 </complexType>
</element>
```

図 2　スキーマ定義

Fig.2　Schema definition.

```
<CreditCardInformation>
  <CreditCardAuthority>XYZ</CreditCardAuthority>
  <CreditCardNumber>0123456789</CreditCardNumber>
  <ExpireDate>2008-12</ExpireDate>
  <CardHolderName>Nakayama Kojiro</CardHolderName>
</CreditCardInformation>
```

図3　XMLデータ
Fig.3　XML data.

多くのXMLプロセッサではXMLデータに対するスキーマ検証の機能を提供している．スキーマ検証には，次に示す(1)XML データが妥当であることを確認する処理と，(2)XML データに情報を追加し PSVI（Post-Schema-Validation Infoset）を生成する処理，の2つが含まれる．

(1) 妥当であることの確認

XML プロセッサは，スキーマ検証において XML データがスキーマで定義されている制約条件を満たしていることを確認する．これにより，スキーマに従っていない想定外の XML データを除外することできるため，アプリケーションの信頼性向上やセキュリティリスクの軽減といった効果が期待できる．特にインターネット上から XML データを受信する場合や不特定多数のシステムから XML データを受信する場合など，XML データの信頼性が低い場合には妥当の確認は非常に重要である．

(2) PSVI の生成

XML Schema を用いてスキーマ検証を行なう場合，XML データの妥当性を確認する処理に加え，XML Schema に含まれる情報を XML データに追加し PSVI を生成する処理を行なう．PSVI に追加される情報には，例えば XML Schema 内に記述されているデフォルト値やデータ型の情報などがある．例えば複数の XML データが共通の属性値を持つ場合，属性を XML データ内に記述せずにデフォルト値として XML Schema 内に記述することで XML データのデータ量を減らすことができる．また，PSVI にデータ型の情報を含めることで，アプリケーションでデータ型を利用した検索等の処理を実行することができる．なお，スキーマ検証により PSVI を生成することに関しては賛否両論があるが[13][14]，本稿では PSVI の是非については論じない．

2.3. XML 暗号

XML 暗号[8]は，任意のデータに対する暗号化の処理方法と暗号化された結果を XML で表現する方法を規定した標準仕様である．XML暗号を用いて XML データに対して暗号化を行う場合，XML データの一部分だけを部分的に暗号化することができる．XML 暗号の仕様では XML データに対する暗号化の方法として，要素に対する暗号（以下，エレメント暗号と呼ぶ）と要素内容に対する暗号（以下，コンテント暗号と呼ぶ）の2種類の方法を規定している．図3に示した XML データに対して，エレメント暗号により暗号化を行った例を図4に，コンテント暗号により暗号化を行なった例を図5に示す．ここでは，CreditCardInformation 要素の子要素である CreditCardNumber 要素に対して部分暗号化を行っている．

エレメント暗号では暗号化対象要素全体の暗号化を行なう．図4に示した暗号化後のデータでは，暗号化対象要素である CreditCardNumber 要素が EncryptedData 要素に置換されている．EncryptedData 要素は XML暗号の仕様で定義されている要素であり，暗号化された結果に関するデータを表す．EncryptedData 要素の満たすべき制約条件は XML 暗号のスキーマによって定義されている．ここでは説明を簡単にするため，EncryptedData 要素の子要素として暗号化結果のデータを表す CipherData 要素のみを持つものとしたが，XML 暗号の仕様では他にも暗号アルゴリズムに関する情報や復号に使用する鍵に関する情報などを記述する方法も定めている．

一方，コンテント暗号では暗号化対象要素の要素内容の暗号化を行なう．図5に示した例では，暗号化対象である CreditCardNumber 要素の要素内容が暗号化されており，EncryptedData 要素は CreditCardNumber 要素の子要素として追加されている．

```
<CreditCardInformation>
  <CreditCardAuthority>XYZ</CreditCardAuthority>
  <xenc:EncryptedData
      Type="http://www.w3.org/2001/04/xmlenc#Element"
    <xenc:CipherData>
      <xenc:CipherValue>fhRzmys1...</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
  <ExpireDate>2008-12</ExpireDate>
  <CardHolderName>Nakayama Kojiro</CardHolderName>
</CreditCardInformation>
```

図4　エレメント暗号
Fig.4　Element encryption.

```
<CreditCardInformation>
  <CreditCardAuthority>XYZ</CreditCardAuthority>
  <CreditCardNumber>
    <xenc:EncryptedData
        Type="http://www.w3.org/2001/04/xmlenc#Content"
      <xenc:CipherData>
        <xenc:CipherValue>eP4Inq2t...</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </CreditCardNumber>
  <ExpireDate>2008-12</ExpireDate>
  <CardHolderName>Nakayama Kojiro</CardHolderName>
</CreditCardInformation >
```

図5　コンテント暗号

Fig.5　Content encryption.

以下では，暗号化前の XML データが従っていたスキーマをオリジナルスキーマと呼ぶ．上で説明したように XML 暗号を用いて部分暗号化を行なうことで XML データのデータ構造が大きく変わるため，暗号化後の XML データはオリジナルスキーマに対する妥当性が保証されない．オリジナルスキーマにおいて暗号化前の XML データと暗号化後の XML データの両方を許すようなスキーマ定義をしている場合を除き，暗号化後の XML データはオリジナルスキーマに対して妥当性が失われる．

3. 課題と研究の目的

2.1節で述べたように，三者間 Web サービスにおいて end-to-end の秘匿性を確保する場合，中継者では暗号化されたままの状態の XML データを処理する必要がある．暗号化された XML データはオリジナルスキーマに対する妥当性が保証されていないため，中継者ではスキーマ検証を実施することができない．

2.2節で述べたように，スキーマ検証の処理には XML データの妥当性を確認する処理と PSVI の生成処理とが含まれる．中継者では XML データの妥当性が確認できないため，アプリケーションの信頼性低下やセキュリティリスクの増大といった問題が発生する．また，PSVI の生成処理ができないため，アプリケーションが PSVI を前提としている場合には想定外の動作をする可能性がある．

本研究の目的は，上記の問題点を解決し end-to-end に渡るセキュアな Web サービスを実現することにある．そこで本研究では，暗号化に対応したスキーマを自動生成することにより暗号化 XML データのスキーマ検証を可能にするスキーマ検証方式を提案する．

4. 提案方式

4.1. スキーマ検証の処理内容

暗号化 XML データのスキーマ検証として，次の3つの処理が考えられる．本稿ではこれらの全ての処理を実施可能なスキーマ検証方式を提案する．

(1) 非暗号化部分のスキーマ検証

暗号化 XML データは，暗号化部分（EncryptedData 要素）と非暗号化部分（EncryptedData 要素以外の部分）から構成される．

非暗号化部分はオリジナルスキーマに従って構成されているため，オリジナルスキーマに基づいてスキーマ検証を行なうことができる．非暗号化部分に含まれるデータは，多くの場合，中継者のアプリケーションで使用されるデータである．よって，非暗号化部分のスキーマ検証を行なうことで，アプリケーションで使用するデータの妥当性が確認でき，信頼性の向上やセキュリティリスクの軽減が期待できる．

(2) 暗号化部分のスキーマ検証

暗号化部分は XML 暗号のスキーマに従って構成されているため，XML 暗号のスキーマに基づいてスキーマ検証を行なうことができる．暗号化部分のスキーマ検証を行なうことで，受信した XML データが XML 暗号に従い正しく暗号化が行なわれていることを確認することができる．

(3) 暗号化箇所の確認

非暗号化部分及び暗号化部分のスキーマ検証に加え，暗号化箇所の確認処理がある．暗号化が必要な箇所が確実に暗号化されていること，暗号化されていてはいけない箇所が暗号化されていないことを確認することができる．

4.2. 提案方式の概要

本稿で提案するスキーマ検証方式の概要を図6に示す．本提案方式では，予め保持しているオリジナルスキーマを変換し暗号化後の XML データが従うスキーマ（ポスト暗号化スキーマ）を生成する．ポスト暗号化スキーマを用いてスキーマ検証を行なうことで，4.1節で述べた暗号化 XML データのスキーマ検証処理を実現する．

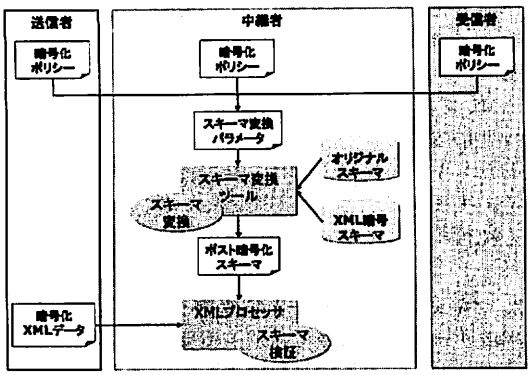

図 6　スキーマ検証方式の概要

Fig.6　Overview of the proposed method.

本提案方式では次の順に処理を行なう．

(1)　スキーマ変換パラメータの決定

ポスト暗号化スキーマを生成するために必要となる暗号化に関するパラメータを決定する．以下では，ポスト暗号化スキーマの生成に必要となる暗号化に関する情報をスキーマ変換パラメータと呼ぶ．スキーマ変換パラメータは各システムが持つセキュリティポリシーなどの情報から決定される（詳細は4.3節を参照）．

(2)　ポスト暗号化スキーマ生成

次に，オリジナルスキーマを変換することでポスト暗号化スキーマを生成する．スキーマ変換は，(1)で決定したスキーマ変換パラメータに基づいて実施される（詳細は4.4節を参照）．

(3)　スキーマ検証

(2)で生成したポスト暗号化スキーマを用いてスキーマ検証を実行する．スキーマ検証の処理自体は従来通りの方法を用いるため，既存の XML プロセッサを利用して実施することが可能である．

4.3. スキーマ変換パラメータの決定

ポスト暗号化スキーマを生成するために必要となる，Web サービス実行時に適用される暗号化に関する情報（スキーマ変換パラメータ）を決定する．スキーマ変換パラメータには次に示す(1)対象要素，(2)要求レベル，(3)暗号タイプの情報が含まれる．

(1)　対象要素

暗号化対象となる要素．エレメント暗号の場合，ここで指定した要素全体が暗号化されることを表す．コンテント暗号の場合，ここで指定した要素の要素内容が暗号化されることを表す．

(2)　要求レベル

暗号化に関する要求のレベル．要求レベルは MUST もしくは MAY を指定する．MUST が指定された場合，対象要素が必ず暗号化されていなければならない．MAY が指定された場合，対象要素の暗号化は任意であるとする．すなわち，対象要素が暗号化されていてもされていなくてもよい．

(3)　暗号タイプ

XML 仕様で規定されている暗号化のタイプ．2.3 節で説明したようにエレメント暗号とコンテント暗号の２種類がある．

4.4. ポスト暗号化スキーマの生成

本提案方式では，スキーマ変換ツールを用いてポスト暗号化スキーマを生成する．スキーマ変換ツールは，暗号化後の XML データが従うべきスキーマを自動生成する．その際，スキーマ変換パラメータとして指定した要求レベル及び暗号タイプに応じて次のようなスキーマ変換処理を行なう．

要求レベルが MUST の場合，暗号化後の XML データでは暗号化対象が EncryptedData 要素に置換されていなければならない．そのためスキーマ変換では，オリジナルスキーマの暗号化対象が出現する箇所に EncryptedData 要素が出現するようにスキーマを変換する．一方，要求レベルが MAY の場合，暗号化後の XML データでは暗号化対象が EncryptedData 要素に置換されているか，置換されずにそのままの状態で出現するかのどちらかである．そのため，要求レベルが MAY の場合は，オリジナルスキーマの暗号化対象が出現する箇所に EncryptedData 要素もしくは暗号化対象が出現するようにスキーマ変換を行なう．

また，暗号タイプがエレメント暗号の場合，対象要素が出現する箇所に EncryptedData 要素が出現するようにスキーマ変換を行なう．一方，暗号タイプがコンテント暗号の場合，対象要素の定義を変更し対象要素の子要素として EncryptedData 要素が出現するようにスキーマ変換を行なう．

図２に示した XML Schema によるスキーマ定義を変換して生成したポスト暗号化スキーマの例を図７に示す．ここでは，スキーマ変換パラメータとして次の値を用いた．

・対象要素　：CreditCardNumber 要素
・要求レベル：MAY
・暗号タイプ：エレメント暗号

要求レベルが MAY であるため，オリジナルスキーマにおいて暗号化対象である CreditCardNumber 要素が出現する箇所が，ポスト暗号化スキーマでは XML Schema の choice 要素に置換されている．choice 要素の子要素として CreditCardNumber 要素と EncryptedData 要素を記述することで，CreditCardNumber 要素と

EncryptedData 要素のどちらかが出現することを表す．また，暗号タイプがエレメント暗号であるため，対象要素が出現する箇所（CreditCardInformation 要素の定義内）に EncryptedData 要素が出現するようにスキーマを変更している．(暗号タイプがコンテント暗号の場合は，対象要素である CreditCardNumber 要素の定義を変更し CreditCardNumber 要素の子要素として EncryptedData 要素が出現するようにする．)

```
<xs:element name="CreditCardInformation">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="CreditCardAuthority" minOccurs="0" />
   <xs:choice minOccurs="0">
    <xs:element ref="CreditCardNumber" />
    <xs:element ref="xenc:EncryptedData" />
   </xs:choice>
   <xs:element ref="ExpireDate" minOccurs="0" />
   <xs:element ref="CardHolderName" minOccurs="0" />
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

図 7　ポスト暗号化スキーマ
Fig.7　Post encryption schema.

このようにして生成したポスト暗号化スキーマを用いてスキーマ検証を行なうことで，4.1節で述べた暗号化 XML データのスキーマ検証処理を実現することができる．

5. 考察

5.1. 提案方式の優位性

本稿で提案したスキーマ検証方式の他にも，Web サービスの中継者においてスキーマ検証を可能にする方法はいくつか考えられる．例えば，オリジナルスキーマを設計する際に暗号化を考慮し，暗号化前と暗号化後の XML データの両方を許すスキーマを定義する方法が考えられる．しかし，暗号化のようなセキュリティ要件は使用する状況に応じて変更する可能性があり，スキーマを設計する時点では確定できないことが多い．そのため，オリジナルスキーマを設計する際に暗号化を考慮したスキーマを定義することは一般的には困難である．特に標準仕様として既定されたスキーマ（例えば，旅行業界で用いられる XML データのフォーマットを規定した TravelXML[15]など）では，多くのシステムで幅広く使用することを想定しており，暗号化などのセキュリティ要件は考慮されていない．本稿で提案したスキーマ検証方式を用いることで，暗号化が考慮されていないスキーマを使用する場合でもスキーマ検証を実施することができる．

また，他のスキーマ検証の方法として独自のスキーマ検証ロジックを実装し，XML プロセッサに組み込む方法が考えられる．しかし，この方法を用いた場合，XML プロセッサごとに異なる対応が必要になる．現在，XML パーサ，データバインディングツールなど，様々な XML プロセッサが存在するが，利用する XML プロセッサに応じて個別に対応する必要がある．本稿で提案したスキーマ検証方式は，全ての XML プロセッサをそのまま利用することができるため，XML プロセッサの種類に応じて個別に対応する必要はない．

5.2. セキュリティポリシーとの連携

Web サービスの暗号化に関する要件はセキュリティポリシーとして記述，公開することがある．セキュリティポリシーを記述するための仕様としては，例えば WS-SecurityPolicy[16]がある．暗号化に関する要件がセキュリティポリシーが公開されている場合，スキーマ変換パラメータはセキュリティポリシーから自動的に決定できる可能性がある．セキュリティポリシーからスキーマ変換パラメータを決定する方法については今後の課題である．

6. おわりに

本稿では Web サービスの中継者において適用可能な暗号化 XML データのスキーマ検証方式を提案した．本提案方式では，暗号化後の XML データが従うスキーマ（ポスト暗号化スキーマ）を生成し，ポスト暗号化スキーマによりスキーマ検証を行なうことで，暗号化 XML データのスキーマ検証を実現している．これにより Web サービスの中継者においてスキーマ検証を実施することが可能になる．

謝辞

本研究は，XML コンソーシアム sPat プロジェクトにおいて検討した内容をまとめたものです．sPat プロジェクトに参加して頂いた各社（アドソル日進（株），（株）内田洋行，キヤノン（株），（株）JIEC，東京エレクトロン(株)，日本電気(株)，(株)ネットタイム，（株）ノムラシステムコーポレーション，(株)日立製作所，PFU アクティブラボ(株)，富士ゼロックス(株)，富士通（株)）の皆様に心から感謝いたします．

文　献

[1] 田中哲雄，湯本真樹，齋礼，“企業情報システムにおける連携技術，” 電学論 C，Vol. 124，No. 5，pp.1051-1057，2004.

[2] D.Krafzig, K.Banke, and D.Slama, "Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series)," Prentice Hall Ptr, 2004.

[3] “Simple Object Access Protocol (SOAP) 1.1,” W3C Note, 2002, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[4] "Security Challenges, Threats and Countermeasures Version 1.0," WS-I Final Material, 2005, http://www.ws-i.org/Profiles/BasicSecurity/Security Challenges-1.0.pdf

[5] "Web Services Security: SOAP Message Security 1.0 (WS-Security 2004), " OASIS Standard, 2004, http://docs.oasis-open.org/wss/2004/01/oasis-200401 -wss-soap-message-security-1.0.pdf

[6] K.Nakayama, T.Ishizaki, and M.Oba, "Application of Web Services Security Using Travel Industry Model," SAINT 2005 Workshop, pp 358-361, 2005.

[7] 中山弘二郎，植田良一，大場みち子，"XML 暗号による部分暗号化後のスキーマ検証方法，" 電気学会 情報システム研究会，IS-06-06，2006.

[8] "XML Encryption Syntax and Processing," W3C Recommendation, 2003, http://www.w3.org/TR/2002/REC-xmlenc-core-20021 210/

[9] "Extensible Markup Language (XML) 1.0 (Third Edition)," W3C Recommendation, 2004, http://www.w3.org/TR/2004/REC-xml-20040204/

[10]"XML Schema Part 1: Structures Second Edition," W3C Recommendation, 2004, http://www.w3.org/TR/2004/REC-xmlschema-1-2004 1028/

[11]"XML Schema Part 2: Datatypes Second Edition," W3C Recommendation, 2004, http://www.w3.org/TR/2004/REC-xmlschema-2-2004 1028/

[12]"RELAX NG Specification," OASIS Committee Specification, 2001, http://www.oasis-open.org/committees/relax-ng/spec-20011203.html

[13]村田真，"XML 誕生秘話：形式言語理論と XML，" 情報処理，44 巻 7 号，pp.743-746，2003.

[14]A.Gregory, and E.Gutentag, "UBL and Object-Oriented XML: Making Type-Aware Systems Work," XML CONFERENCE & EXPOSITION 2003, 2003, http://www.idealliance.org/papers/dx_xml03/papers/0 4-04-04/04-04-04.html

[15] "TravelXML 仕様公開ページ，" XML コンソーシアム，http://www.xmlconsortium.org/wg/TravelXML/Trave lXML_index.html

[16] "Web Services Security Policy Language (WS-SecurityPolicy), " 2005, http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ ws-securitypolicy.pdf