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

2007－ＤＤ－62（13）
2007／7／27

# 暗号化XMLデータのスキーマ検証のための ポスト暗号化スキーマ生成方法

中山 弘二郎† 大場 みち子†† 薦田憲久†††

† （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
†† （株）日立製作所 ソフトウェア事業部 〒244-8555 神奈川県横浜市戸塚区戸塚町 5030
††† 大阪大学大学院 情報科学研究科 〒565-0871 大阪府吹田市山田丘 2-1

E-mail: † kojiro.nakayama.fc@hitachi.com, †† michiko.oba.cq@hitachi.com,
††† komoda@ist.osaka-u.ac.jp

あらまし XML暗号によりXMLデータの部分暗号化を行う場合，暗号化後のデータはオリジナルのスキーマに対する妥当性が保証されないためスキーマ検証を実施することができない．暗号化後のXMLデータに対してスキーマ検証を行なう方法として，暗号化後のデータが妥当となるようなスキーマ（ポスト暗号化スキーマ）を作成する方法が考えられるが，変換の影響範囲の問題やUPA（Unique Particle Attribution）違反の問題が発生するため，ポスト暗号化スキーマの生成は容易ではない．本稿では，上記の問題を解決したポスト暗号化スキーマの生成方法を提案する．提案方式では，ローカル展開及びパーティクル統合により上記の問題を解決する．

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

# A Method for Generating a Post-Encryption Schema for a Schema Validation of Encrypted XML Data

Kojiro NAKAYAMA† Michiko OBA†† and Norihisa KOMODA†††

† Systems Development Laboratory, Hitachi, Ltd. 1099 Ohzenji, Asao-ku, Kawasaki-shi, Kanagawa 215-0013 Japan
†† Software Division, Hitachi, Ltd. 5030 Totsuka-cho, Totsuka-ku, Yokohama-shi, Kanagawa 244-8555 Japan
††† Graduate School of Information Science and Technology, Osaka University 2-1 Yamadaoka, Suita-shi, Osaka 565-0871 Japan

E-mail: † kojiro.nakayama.fc@hitachi.com, †† michiko.oba.cq@hitachi.com,
††† komoda@ist.osaka-u.ac.jp

**Abstract** If XML data is partially encrypted using XML Encryption, we cannot perform shcema validation because the data is no longer valid against the original schema. To perform schema validation, it is required to provide a post-encryption schema. However, there are problems in generating a post-encryption schema such as the scope of impact and violation of UPA constraint.In this paper, we propose a method for generating a post-encryption schema. We used local expansion and particle composition to avoid impacting the element declaration beyond the intended extent and to avoid violating the UPA constraint.

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

## 1. はじめに

近年，XML（Extensible Markup Language）[1]ベースの標準技術を利用したシステム連携技術であるWebサービスが注目を集めている．Webサービスでは，システム間でSOAP（Simple Object Access Protocol）[2]により規定されたメッセージ（SOAPメッセージ）を交換することでシステム連携を行なう．SOAPメッセージは中継者を介して最終的な受信者まで転送されることがある．このようなマルチホップなWebサービスにおいて，最初の送信者から最終的な受信者までのEnd-to-Endの秘匿性を確保するには，メッセージレベルで暗号化を行なう必要がある[3]．メッセージレベルでの暗号化を実現するための技術としてXML暗号[4]がある．XML暗号を利用するとXMLデータの一部分を部分暗号化することができるため，SOAPメッセージ内のデータのうち中継者に開示したくない部分だけを部分的に暗号化し，最初の送信者と最終的な受信者の間でのみ秘密を共有することができる．しかし，こ

のようにXML暗号を用いてXMLデータの部分暗号化を行なう場合，暗号化後のデータは元々のスキーマ定義に対する妥当性が失われる可能性がある．そのため，中継者では暗号化データのスキーマ検証を実施することができないという問題が発生する[5][6]．

暗号化データのスキーマ検証を実施する方法として，暗号化後のデータが妥当となるようなスキーマ(ポスト暗号化スキーマ）を提供する方法が考えられる[7][5][6]．しかし，変換の影響範囲の問題や変換後のスキーマにおけるUPA（Unique Particle Attribution）違反の問題のため，ポスト暗号化スキーマの生成は容易ではない．

本稿では，上記の問題点を解決したポスト暗号化スキーマの生成方法を提案する．提案方式では，ローカル展開及びパーティクル統合の処理を行なうことで影響範囲の問題及びUPA違反の問題を解決する．

## 2. 暗号データのスキーマ検証

### 2.1. XML Schema

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のXMLデータが満たすべき制約条件は，スキーマ定義により記述される．XMLデータがスキーマ定義の持つ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．Webサービスでは，ネットワークを介してシステム間でXMLデータを交換するため，スキーマ検証によりXMLデータの妥当性を検証することが望ましい．特に，インターネット上にサービスを公開する場合には，様々な攻撃を受けるリスクがあるため，スキーマ検証により攻撃のリスクを低減することが重要である．

XMLのスキーマ定義を記述する言語には，DTD[1]，XML Schema[8]，Relax NG[9]などがある．Webサービスでは，SOAPメッセージに含まれるデータのスキーマはXML Schemaを用いて定義することが推奨されているため[10]，本稿ではスキーマ言語としてXML Schemaを対象とする．XML Schemaを用いたスキーマ定義の例を図1に示す（記述を簡単にするため一部省略している．以下の例も同様）．また，図1に示したスキーマ定義に対して妥当であるXMLデータの例を図2に示す．

```
<xs:element name="order">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="applicant" />
      <xs:element ref="item" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="applicant">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
      <xs:element name="email" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="item">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
      <xs:element name="code" type="xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

図1　XML Schema定義

Fig. 1.　XML schema definition.

```
<order xmlns="http://example.org/order">
  <applicant>
    <name>John</name>
    <email>john@example.com</email>
  </applicant>
  <item>
    <name>Laptop Computer</name>
    <code>14048135</code>
  </item>
</order>
```

図2　XMLデータ

Fig. 2.　XML data.

### 2.2. XML暗号

XML暗号[4]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タイプとして，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種類を規定している．図2に示したXMLデータのapplicant要素に対して，エレメント暗号により暗号化を行った結果を図3に，コンテント暗号により暗号化を行なった結果を図4に示す．

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3に示した例では，暗号化対象要素であるapplicant要素全体が暗号化され，EncryptedData要素に置換されている．EncryptedData要素は暗号化された結果を表す要素であり，EncryptedData要素のスキーマはXML暗号の仕様により規定されている．

```
<order xmlns="http://example.org/order">
  <xenc:EncryptedData
    xmlns:xenc="http://..." Type="http://...#Element">
    <xenc:EncryptionMethod ...>
    <xenc:CipherData ...>
  </xenc:EncryptedData>
  <item>
    <name>Laptop Computer</name>
    <code>14048135</code>
  </item>
</order>
```

図3　エレメント暗号

Fig. 3.　Element encryption.

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4に示した例では，暗号化対象である applicant 要素の要素内容が暗号化されており，EncryptedData 要素は applicant 要素の子要素として追加されている．

```
<order xmlns="http://example.org/order">
  <applicant>
    <xenc:EncryptedData
      xmlns:xenc="http://..." Type="http://...#Content">
      <xenc:EncryptionMethod ...>
      <xenc:CipherData ...>
    </xenc:EncryptedData>
  </applicant>
  <item>
    <name>Laptop Computer</name>
    <code>14048135</code>
  </item>
</order>
```

図4　コンテント暗号

Fig. 4.　Content encryption.

### 2.3. 暗号化データのスキーマ検証

XML 暗号による暗号化を行なうと，暗号対象要素もしくはその子要素が EncryptedData 要素に置換されるため，暗号化後の XML データは暗号化前の XML データが従っていたスキーマ（以下，オリジナルスキーマと書く）に対する妥当性が失わる．そのため，暗号化後の XML データはオリジナルスキーマを用いてスキーマ検証が実施することができない[5][6].

暗号化後の XML データに対してスキーマ検証を行なう方法として，暗号化後の XML データが従うようなスキーマ（以下，ポスト暗号化スキーマと書く）を用いる方法がある[7][5][6]．しかし，ポスト暗号化スキーマを生成する際には変換の影響範囲の問題や変換後のスキーマにおける UPA（Unique Particle Attribution）違反の問題が発生する．これらの問題の詳細は次の章で述べる．

## 3. ポスト暗号化スキーマ生成における課題

### 3.1. 表記方法

本稿ではスキーマ定義の記述を簡単化するため，スキーマ定義を構成する要素（スキーマコンポーネント）を次のように表記する．

（1）要素宣言の表記

要素宣言（Element Declaration）とは，要素名と型定義とを関連付けるスキーマコンポーネントである．本稿では，暗号化対象要素の要素宣言を Et，EncryptedData 要素の要素宣言への参照を Ee を用いて表記する．

Ee 及び Et に続く中括弧は要素の出現回数を表す．Et{a,b}は暗号対象要素の要素宣言を表し，その最小出現回数（minOccurs 属性の値）が a であり，最大出現回数（maxOccurs 属性の値）が b であることを表す．また，例えば Ee{3,5}は次のような要素宣言を表す．

```
<xsd:element
    ref="xenc:EncryptedData">
    minOccurs="3" maxOccurs="5" />
```

（2）モデルグループの表記

モデルグループ（Model Groups）とは，要素の出現順に関する制約を表すスキーマコンポーネントである．本稿では，文字列" sequence"," choice" を用いてモデルグループを表す．

sequence{a,b}(P1,P2)は，次のようなモデルグループを表す．ここで，P1，P2 は任意のスキーマコンポーネントを表す．

```
<xsd:sequence
    minOccurs="a" maxOccurs="b">
    P1,P2
</xsd:sequence>
```

同様に，choice{a,b}(P1,P2)は次のようなモデルグループを表す．

```
<xsd:choice
    minOccurs="a" maxOccurs="b">
    P1,P2
</xsd:choice>
```

### 3.2. ポスト暗号化スキーマの生成

本節では，オリジナルスキーマからポスト暗号化スキーマを生成する方法を述べる．後述するように，本節で述べる方法には問題があり，適切なポスト暗号化スキーマを生成できないことがある．これらの問題を解決する方法は4章で述べる．

ポスト暗号化スキーマの生成の概要を図5に示す．オリジナルスキーマからポスト暗号化スキーマを生成するためには，パラメータとして暗号化の要件に関する次の3つの情報を与える必要がある．

・**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．対象要素の指定には XML データに対する XPath 式を用いて記述する．

・**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り，必ずしも暗号化されていなくてもよい．

・**暗号タイプ**：XML 仕様で規定されている暗号化のタイプ．前述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．

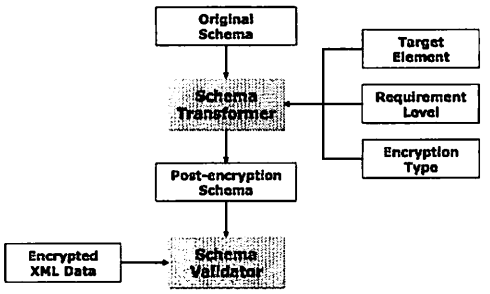


図5　システムアーキテクチャ

Fig. 5.　System architecture.

### 3.3. スキーマ変換

ポスト暗号化スキーマとは，暗号対象データが出現する代わりに暗号化の結果である EncryptedData 要素が出現することを許すようなスキーマである．ポスト暗号化スキーマの生成は，オリジナルスキーマにおける暗号対象要素の要素宣言を特定し，特定した要素宣言を暗号化に対応する形式に変換することで実現される．

要素宣言の特定には，パラメータとして与える対象要素の情報を用いる．対象要素として，例えば，次のような値をパラメータとして与える．

```
/order/applicant
```

この場合，order 要素の子要素である applicant 要素が暗号化されることを表す．要素宣言の特定処理では，対応する次の要素宣言が Et として特定される．

```
<xs:element name="applicant">
    (省略)
</xs:element>
```

次に，特定した暗号化対象要素の要素宣言に対して暗号化に対応するように置換処理を行なう．要素宣言に対する置換処理の内容は，パラメータとして指定する要求レベル，暗号タイプによって異なる．次に，それぞれのパラメータを与えた際の変換処理について説明する．

（1）MUST／エレメント暗号

要求レベルが MUST，暗号タイプがエレメント暗号の場合，次に示す変換を行なう．

$$E_t\{a,b\} \rightarrow E_e\{a,b\}$$

MUST／エレメント暗号の場合，暗号化後の XML データにおいて対象要素の代わりに EncryptedData 要素が出現しなくてはならない．そのため，スキーマ定義においても Et を Ee に置換する．

（2）MAY／エレメント暗号

要求レベルが MAY，暗号タイプがエレメント暗号の場合，次に示す変換を行なう．

$$E_t\{a,b\} \rightarrow choice\{a,b\}(E_e, E_t)$$

MAY／エレメント暗号の場合，暗号化後の XML データにおいて対象要素と EncryptedData 要素のどちらかが出現する．そのため，スキーマ定義では対象要素の宣言を choice に置換し，choice 内に対象要素と EncryptedData への参照を記述する．

（3）MUST／コンテント暗号

要求レベルが MUST，暗号タイプがコンテント暗号の場合，次に示す変換を行なう．

$$E_t\{a,b\} \rightarrow E_t'\{a,b\}(sequence(E_e))$$

ここで Et' は，Et と同じ名前と属性を持つが，型の異なる要素宣言を表す．MUST／コンテント暗号の場合，暗号化後の XML データにおいて対象要素の子要素として EncryptedData 要素が出現することになる．そのため，スキーマ定義では対象要素の子要素として EncryptedData 要素が出現するように置換する．

（4）MAY／コンテント暗号

要求レベルが MAY，暗号タイプがコンテント暗号の場合，次に示す変換を行なう．

$$E_t\{a,b\} \rightarrow E_t'\{a,b\}(choice(E_e, content(E_t)))$$

ここで，関数 content(Et)は，要素宣言 Et の子要素の集合を表すものとする．MAY／コンテント暗号の場合，暗号化後の XML データにおいて対象要素の子要素として，EncryptedData 要素もしくはオリジナルスキーマにおける対象要素の子要素が出現することになる．そのため，スキーマ定義では対象要素の型が choice に置換し，choice 内に EncryptedData への参照と対象要素の子要素を記述する．

### 3.4. ポスト暗号化スキーマ生成における問題

上述したポスト暗号化スキーマの生成方法には，置

換の影響範囲の問題とUPA違反の問題があるため，スキーマ定義の記述方法や暗号化の要件によっては正しく置換処理を行なうことができない．本節ではこの2つの問題について説明する．

（1）置換の影響範囲の問題

XML Schemaは要素宣言や型定義への参照メカニズムを提供している．参照メカニズムを利用することで，一つの要素宣言や型定義が複数個所から利用することができ，スキーマコンポーネントの再利用性や保守性が高まる．スキーマ定義における要素宣言の参照関係の例を図6に示す．

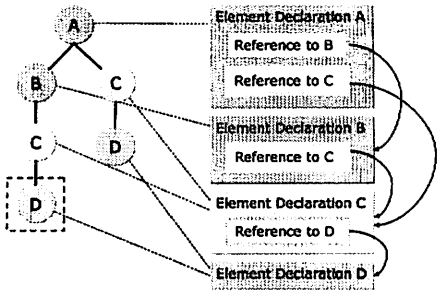


図6　スキーマ定義における参照関係

Fig. 6.　Reference mechanism in a schema definition.

このような参照メカニズムを利用したスキーマ定義に対して3.3節で述べた置換処理を行なう場合，意図した範囲以外の箇所にも置換の影響が及ぶ可能性がある．例えば，パス"/A/B/C/D"を満たすD要素だけを暗号化したい場合，単純にDの要素宣言を置換するとパス"/A/C/D"を満たすD要素にも影響が及んでしまう．その結果，生成されたポスト暗号化スキーマでは，パス"/A/C/D"を満たすD要素が暗号化されていない場合，妥当ではないと判断されてしまう可能性がある．

（2）UPA違反の問題

XML Schemaの仕様では，XMLデータにおける要素が曖昧さなくスキーマ定義上のパーティクル（要素宣言やモデルグループなどのスキーマコンポーネント）と対応付けられなくてはならないとしている．この制約条件はUPAと呼ばれる．

3.3節で述べたように，ポスト暗号化スキーマ生成の際の置換処理によって暗号対象要素の要素宣言はEncryptedData要素への参照を含むスキーマコンポーネントに置換される．そのため，暗号化対象要素が複数存在する場合，ポスト暗号化スキーマには複数のEncryptedData要素への参照が含まれることになり，UPA違反が発生する可能性がある．

図7にUPAに違反したスキーマ定義の例を示す．このスキーマ定義には二つのEncryptedData要素への参照（Ee）が含まれている．最初のEeの最小出現回数が0であるため，XMLデータ上のEncryptedData要素がどちらのEeに対応しているかを決定することができない．そのため，図7に示したスキーマ定義はUPA違反であるとされ，正しいスキーマ定義とは見做されない．

```
<xs:element name="root">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="xenc:EncryptedData" minOccurs="0"/>
      <xs:choice>
        <xs:element ref="xenc:EncryptedData" />
        <xs:element ref="b" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

図7　UPA違反

Fig. 7.　Violation of UPA constraint.

このように暗号対象要素が複数存在する場合，たとえ暗号化前は異なる名前の要素宣言であっても，暗号化によって両方がEncryptedData要素に置換されるため，UPA違反が発生することがある．

## 4. ポスト暗号化スキーマ生成方法の提案

### 4.1. 提案方法の概要

本章では，3.4節で述べた問題を解決した新しいポスト暗号化スキーマ生成方法を提案する．本方式では，置換の影響範囲の問題を解決するため，最初にローカル展開の処理を行なう．ローカル展開の詳細は4.2節で述べる．次に3.3節で述べた要素宣言の置換処理を行なう．最後に，UPA違反の問題を解決するため，パーティクル統合の処理を行なう．パーティクル統合の詳細は4.3節で述べる．

### 4.2. ローカル展開

本提案方式では，変換処理を実行する前にグローバル要素宣言及びグローバル型定義をローカルに展開することで，他の箇所に影響が及ぶのを防ぐ．ローカル展開では，スキーマ内の全ての要素宣言に対して次の処理を実行する．

・要素宣言がref属性を持つ場合，処理対象の要素宣言を参照先の要素宣言に変換する．

・要素宣言がtype属性を持つ場合，参照先の型定義を，対象要素宣言の匿名型として追加する．

ローカル展開により，グローバル要素やグローバル型定義を参照することがなくなるため，変換処理により意図しない箇所へ影響が及ぶことを防ぐことができる．

### 4.3. パーティクル統合

本提案方式では，3.3節で述べたスキーマ変換によりUPA 違反が発生した場合，パーティクル統合によりUPA違反の問題を解決する．パーティクル統合の処理は，パーティクルが含まれるモデルグループの種類によって異なる．ここでは，パーティクルがsequenceモデルグループに含まれる場合について述べる．

sequenceモデルグループ内では，連続するパーティクルが同じ要素宣言を参照しており，かつ最初のパーティクルの出現回数が固定でない場合，UPA違反が発生する．そのため，sequenceモデルグループ内の連続する複数の要素が暗号対象となっている場合，UPA違反が発生する可能性がある．以下では，sequenceモデルグループ内連続する２つの要素が暗号対象となっている場合のパーティクル統合について述べる．

（１）MUST／MUST

sequence内の連続する２つの要素宣言が暗号対象であり，両方とも要求レベルがMUSTの場合，変換結果は次のようになる．

$$E_e\{a_1,b_1\},E_e\{a_2,b_2\}$$

ここでa1<b1の場合，UPA違反が発生する．そこで，連続する２つの要素宣言を１つの要素宣言に統合することでUPA違反を回避することができる．連続する２つの要素宣言は次のようにして統合することができる．

$$E_e\{a_1,b_1\},E_e\{a_2,b_2\}\rightarrow E_e\{a_1+a_2,b_1+b_2\}$$

（２）MAY／MUST

sequence内の連続する２つの要素宣言が暗号対象であり，最初の要素宣言の要求レベルがMAY，次の要素宣言の要求レベルがMUSTの場合，変換結果は次のようになる．

$$choice\{a_1,b_1\}(E_e,E_{t1}),E_e\{a_2,b_2\}$$

ここでa1<b1の場合，UPA違反が発生する．そこで，choiceと要素宣言を統合することでUPA違反を回避する．その際，オリジナルスキーマの制約条件を保ったまま統合を行なうことはできず，制約条件を緩める必要がある．連続するchoiceと要素宣言は次のように統合することができる．

$$choice\{a_1,b_1\}(E_e,E_{t1}),E_e\{a_2,b_2\}$$
$$\rightarrow choice\{a_1+a_2,b_1+b_2\}(E_e,E_{t1})$$

上記の統合により，オリジナルスキーマの制約条件が緩められ，オリジナルスキーマでは禁止されていたインスタンスが許可される可能性があるため注意が必要である．

## 5. まとめ

本稿では，暗号化XMLデータのスキーマ検証を実現するために，ポスト暗号化スキーマを生成する方法を提案した．提案方式では，暗号要件に関する情報（暗号対象要素，要求レベル，暗号タイプ）を元にオリジナルスキーマの変換を行い，ポスト暗号化スキーマを生成する．スキーマ変換の際には，変換の影響範囲の問題，UPA違反の問題が発生するが，提案方式ではローカル展開，パーティクル統合によりこれらの問題を解決する．


## 謝辞

本研究は，XMLコンソーシアムsPlatプロジェクトにて検討した内容を発展させたものです．本プロジェクトに参加し，有益な議論をしてくださった皆さまに心から感謝いたします．


## 文　献


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

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

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

[4] "XML Encryption Syntax and Processing," W3C Recommendation, 2003, http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/

[5] 中山弘二郎，大場みち子，荒本道隆，鷹田憲久，"マルチホップWebサービスにおける部分暗号化データの処理方式，"電学論C，Vol.127，No.6，pp.951-956，2007

[6] 中山弘二郎，荒本道隆，大場みち子，"暗号化XMLデータのスキーマ検証方式の提案，"情報処理学会研究報告，2006-DD-056，Vol.2006，No.83，pp.31-37，2006

[7] "XML Encryption Requirements," W3C Note 2002, http://www.w3.org/TR/2002/NOTE-xml-encryption-req-20020304

[8] "XML Schema Part 1: Structures Second Edition," W3C Recommendation, 2004, http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

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

[10] "Basic Profile Version 1.1," WS-I Final Material, 2006, http://www.ws-i.org/Profiles/BasicProfile-1.1-2006-04-10.html