Web Services Security:
SOAP Message Security 1.0
(WS-Security 2004)

OASIS Standard 200401, March 2004

日本語訳 2005 年 3 月

日本語訳作成 (Japanese Translation):

XML コンソーシアム セキュリティ部会 (XML Consortium Security SIG)

日本語訳貢献者 (Contributors):

池上 勝美	IKEGAMI, Katsumi	沖電気工業株式会社
横溝 良和	YOKOMIZO, Yoshikazu	キヤノン株式会社
長岡 圭一	NAGAOKA, Keiichi	東京エレクトロン株式会社
松永 豊	MATSUNAGA, Yutaka	東京エレクトロン株式会社
阿部 和子	ABE, Kazuko	東芝ソリューション株式会社
山田 正隆	YAMADA, Masataka	東芝ソリューション株式会社
澤井 真二	SAWAI, Shinji	日本オラクル株式会社
明石 正則	AKASHI, Masanori	日本テレコム株式会社
岡村 和英	OKAMURA, Kazuhide	株式会社ネット・タイム
村垣 委久夫	MURAGAKI, Ikuo	株式会社日立システムアンドサービス
西村 利浩	NISHIMURA, Toshihiro	富士通株式会社

免責事項 (disclaimer notice):

This translated document is provided by XML Consortium as an informational service to the global community. This is an unofficial, non-normative translation of the official document, Web Services Security: SOAP Message Security 1.0 (WS-Security 2004), located at http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf, © copyright OASIS 2002-2004. This translation is published with acknowledgement of and in agreement with terms specified in the OASIS Translation Policy. Neither OASIS nor XML Consortium assume responsibility for any errors contained herein.

本翻訳文書はグローバルなコミュニティへの情報のサービスとして XML コンソーシアム によって提供される。これは http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf にある公式文書 Web Services Security: SOAP Message Security 1.0 (WS-Security 2004), © copyright OASIS 2002-2004 の非公式の、参考的な翻訳である。本翻訳は OASIS Translation Policy に明記されている条項を承知し同意の上で公表されている。OASIS も XML コンソーシアムもここに含まれるいかなる誤りに対しても責任をもたない。

THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL XML CONSORTIUM, BE LIABLE FOR ANY CONSEQUENTIAL, INCIDENTAL, DIRECT, INDIRECT, SPECIAL, PUNITIVE, OR OTHER DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS) ARISING OUT OF THIS DOCUMENT.

XML コンソーシアムは、本書の記載内容に関して、その正確性、商品性、利用目的への適合性等に関して保証するものではなく、特許権、著作権、その他の権利を侵害していないことを保証するものでもありません。本書の利用により生じた損害について、XML コンソーシアムは、法律上のいかなる責任も負いません。

Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)

OASIS Standard 200401, March 2004

Document identifier:
{WSS: SOAP Message Security }-{1.0} (Word) (PDF)

Location:
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0

Errata Location:
http://www.oasis-open.org/committees/wss

Editor:

Anthony	Nadalin	IBM
Chris	Kaler	Microsoft
Phillip	Hallam-Baker	VeriSign
Ronald	Monzillo	Sun

Contributors:

Gene	Thurston	AmberPoint
Frank	Siebenlist	Argonne National Lab
Merlin	Hughes	Baltimore Technologies
Irving	Reid	Baltimore Technologies
Peter	Dapkus	BEA
Hal	Lockhart	BEA
Symon	Chang	CommerceOne
Srinivas	Davanum	Computer Associates
Thomas	DeMartini	ContentGuard
Guillermo	Lao	ContentGuard
TJ	Pannu	ContentGuard
Shawn	Sharp	Cyclone Commerce
Ganesh	Vaideeswaran	Documentum
Sam	Wei	Documentum
John	Hughes	Entegrity
Tim	Moses	Entrust
Toshihiro	Nishimura	Fujitsu
Tom	Rutt	Fujitsu
Yutaka	Kudo	Hitachi

Jason	Rouault	HP
Paula	Austel	IBM
Bob	Blakley	IBM
Joel	Farrell	IBM
Satoshi	Hada	IBM
Maryann	Hondo	IBM
Michael	McIntosh	IBM
Hiroshi	Maruyama	IBM
David	Melgar	IBM
Anthony	Nadalin	IBM
Nataraj	Nagaratnam	IBM
Wayne	Vicknair	IBM
Kelvin	Lawrence	IBM (co-Chair)
Don	Flinn	Individual
Bob	Morgan	Individual
Bob	Atkinson	Microsoft
Keith	Ballinger	Microsoft
Allen	Brown	Microsoft
Paul	Cotton	Microsoft
Giovanni	Della-Libera	Microsoft
Vijay	Gajjala	Microsoft
Johannes	Klein	Microsoft
Scott	Konersmann	Microsoft
Chris	Kurt	Microsoft
Brian	LaMacchia	Microsoft
Paul	Leach	Microsoft
John	Manferdelli	Microsoft
John	Shewchuk	Microsoft
Dan	Simon	Microsoft
Hervey	Wilson	Microsoft
Chris	Kaler	Microsoft (co-Chair)
Prateek	Mishra	Netegrity
Frederick	Hirsch	Nokia
Senthil	Sengodan	Nokia
Lloyd	Burch	Novell
Ed	Reed	Novell
Charles	Knouse	Oblix
Steve	Anderson	OpenNetwork (Sec)
Vipin	Samar	Oracle
Jerry	Schwarz	Oracle
Eric	Gravengaard	Reactivity
Stuart	King	Reed Elsevier
Andrew	Nash	RSA Security
Rob	Philpott	RSA Security
Peter	Rostin	RSA Security
Martijn	de Boer	SAP
Blake	Dournaee	Sarvega
Pete	Wenzel	SeeBeyond
Jonathan	Tourzan	Sony
Yassir	Elley	Sun Microsystems
Jeff	Hodges	Sun Microsystems
Ronald	Monzillo	Sun Microsystems
Jan	Alexander	Systinet

Michael	Nguyen	The IDA of Singapore
Don	Adams	TIBCO
John	Weiland	US Navy
Phillip	Hallam-Baker	VeriSign
Mark	Hays	Verisign
Hemma	Prafullchandra	VeriSign

Abstract:

本規定では、メッセージの完全性と秘匿性を提供するための SOAP メッセージングの拡張を記述する。規定される仕組みは、広範囲のセキュリティ・モデルと暗号技術に適応して利用されることができる。

本規定はまた、セキュリティ・トークンをメッセージ内容に関連付けるための汎用的な仕組みを提供する。セキュリティ・トークンの特定の型は要求されず、規定は拡張可能に設計されている(すなわち、複数のセキュリティ・トークン形式に対応する)。例えば、クライアントはアイデンティティの証明のためにある形式を提供しつつ、特定のビジネスの認可をもっていることを証明するためには他の形式を提供するかもしれない。

加えて、本規定ではバイナリ・セキュリティ・トークンの符号化方法、XML ベースのトークンのための枠組み、および不透明な暗号化された鍵を含める方法について記述する。また、メッセージに含められるトークンの特性を更に記述するために利用されることにできる拡張性をもつ仕組みが含まれる。

Status:

これは OASIS Web Services Security (WSS)技術委員会により考慮のために提出された技術委員会文書である。コメントは編集者へ送付されたい。もし wss@lists.oasis-open.org リストに含まれているならば、コメントはそこに送付されたい。そのリストに含まれていないなら、wss-comment@lists.oasis-open.org リストを購読し、そこにコメントを送付されたい。購読するには、メッセージの本体として"subscribe"という言葉を入れて wss-comment-request@lists.oasis-open.org へ email メッセージを送付のこと。本規定の実装に本質的かもしれない特許開示情報とライセンス条項の提供については、http://www.oasis-open.org/committees/wss/ipr.php にある OASIS Web Services Security Technical Committee (WSS TC)の Intellectual Property Rights の部分を参照のこと。一般的な OASIS IPR 情報は http://www.oasis-open.org/who/intellectualproperty.shtml に見つけることができる。

Table of Contents

1はじめに

本OASIS仕様は、WSS技術委員会(Technical Committee)による有意の新しい作業の結果であり、そして、入力提出文書、Web Service Security (WS-Security) Version 1.0 April 5, 2002およびWeb Services Security Addendum Version 1.0 August 18, 2002に取って代わる。

本規定では、安全な Web サービスを構築するときに、メッセージ内容の完全性と秘匿性を実装するために利用できる SOAP [SOAP11, SOAP12] 拡張の標準一式を提案する。本規定では、この拡張とモジュール一式を "Web Services Security: SOAP Message Security" または "WSS: SOAP Message Security" として参照する。

本規定は柔軟性に富み、PKI、Kerberos、および SSL を含む幅広いセキュリティ・モデルの中で Web サービスを保護するためのベースとして使われるように設計されている。特に、本規定は複数のセキュリティ・トークン形式、複数の信頼ドメイン、複数の署名形式、および複数の暗号技術のサポートを提供する。これらを利用するためのトークン形式とセマンティクスは、関連するプロファイル文書で定義される。

本規定は、メッセージの部分としてセキュリティ・トークンを送信する能力、メッセージの完全性、およびメッセージの秘匿性の 3 つの主要な機構を提供する。これらの機構はそれ自体では Web サービスのための完全なセキュリティ解を提供しない。その代わり、本規定は広範囲のセキュリティ・モデルとセキュリティ技術を利用するために、他の Web サービス拡張と高レベルのアプリケーション固有のプロトコルと一緒に利用されることができるビルディング・ブロックである。

これらの機構は独立に利用されることもできる (例えば、セキュリティ・トークンを渡す) し、または、固く結び付けられた方法で利用されることもできる (例えば、メッセージまたはメッセージの部分に署名し暗号化し、署名と暗号化に利用された鍵と関連するセキュリティ・トークンまたはトークン・パスを提供する)。

1.1 目標と要件

本規定の目標は、アプリケーションが安全な SOAP メッセージの交換を行なうことを可能にすることである。

本規定は一連のセキュリティ・プロトコルを構築するために利用されることのできる機構の柔軟な集合を提供することを意図している。言い換えると、本規定は意図的に明示的な固定のセキュリティ・プロトコルを説明しない。

どのセキュリティ・プロトコルでもそうだが、本規定を利用して構築されたセキュリティ・プロトコルが幅広い攻撃のどれかから脆弱でないことを保証するために顕著な努力が適用されなければならない。本規定の例は、これらの機構の構文を説明することを意図しており、これらの機構を安全な方法で組合せる例を意図していない。

本規定の焦点は、確立されたセッション、セキュリティ・コンテキスト、および/または ポリシー同意を仮定するかもしれないメッセージ・セキュリティを提供する単一メッセージのセキュリティ言語を説明することである。

安全なメッセージ交換をサポートするための要件が下にリストされる。

1.1.1 要件

Web サービス・セキュリティ言語は幅広いセキュリティ・モデルをサポートしなければならない。次のリストが本規定のための鍵となる要件を識別する。

複数のセキュリティ・トークン形式

複数の信頼ドメイン

複数の署名形式

複数の暗号技術

エンド・トゥ・エンドのメッセージ内容セキュリティでありトランスポート層のセキュリティではない。

1.1.2 目標でないもの

次の話題は本文書の範囲外である。

セキュリティ・コンテキストの確立または認証機構

鍵の導出

セキュリティ・ポリシーの公示と交換

信頼がどのように確立され決定されるか。

否認拒否

2記法と用語

本節では、本規定で利用される記法、名前空間および用語を指定する。

2.1 記法

本文書に出てくるキーワード「しなければならない(MUST)」、「してはならない(MUST NOT)」、「要求されている(REQUIRED)」、「することになる(SHALL)」、「することはない(SHALL NOT)」、「する必要がある(SHOULD)」、「しないほうがよい(SHOULD NOT)」、「推奨される(RECOMMENDED)、「してもよい(MAY)」および「選択できる(OPTIONAL)」は RFC2119 で説明されるように解釈される。

抽象データ・モデルを説明するとき、本規定では XML Infoset により利用される記法を利用する。とくに、抽象プロパティ名はつねに角括弧に現れる (例えば、[some property])。

具体的な XML スキーマを説明するとき、本規定では、要素の [children] または [attributes] プロパティの各メンバーは XPath ライクな記法(例えば、/x:MyHeader/x:SomeProperty/@value1) を利用して説明される。 {any} の利用は、要素ワイルドカード (<xs:any/>) の存在を示す。 @{any} の利用は属性ワイルドカード (<xs:anyAttribute/>) の存在を示す。

読者は Internet Security Glossary [GLOS] の用語定義に精通していることを前提とする。

2.2 名前空間

(一般形式 "some-URI" の) 名前空間 URI は RFC 2396 [URI] で定義されるあるアプリケーション依存または文脈依存の URI を表す。本規定の実装により利用されなければ**ならない (MUST)** XML 名前空間 URI は次の通りとする (本規定で利用される要素が様々な名前空間からのものであることに注意すること):

```
http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
```

本規定は一般の SOAP [SOAP11, SOAP12] メッセージ構造及びメッセージ処理モデルと動作するよう設計されており、SOAP のどの版へも適用可能であるべきである。ここでは現在の SOAP 1.1 名前空間 URI が詳しい例を提供されているために利用されているが、本規定の適用可能性を SOAP の単一の版に限定する意図はない。

本文書で利用される名前空間が次の表に示される (簡単のために、例では以下にリストされた前置詞を利用するが、URI を含まない - 以下にリストされたものが仮定される)。

Prefix	Namespace
ds	http://www.w3.org/2000/09/xmldsig#
S11	http://schemas.xmlsoap.org/soap/envelope/
S12	http://www.w3.org/2003/05/soap-envelope

wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd

231 wsse と wsu 名前空間に対して提供される URL は、スキーマ・ファイルを取得するために利
232 用されることができる。

233 2.3 頭文字と省略形

234 次の (参考の) 表は、本文書のための頭文字と省略形を定義する。

Term	Definition
HMAC	Keyed-Hashing for Message Authentication
SHA-1	Secure Hash Algorithm 1
SOAP	Simple Object Access Protocol
URI	Uniform Resource Identifier
XML	Extensible Markup Language

235 2.4 用語

236 以下に定義されるものが、本規定で利用されるセキュリティ用語のための基本定義である。

237 **申告** - 申告は実体により作られる宣言である (例えば、名前、身元、鍵、グループ、特権、能
238 力など)。

239 **申告の確認** - 申告の確認は実体に適用される申告を検証する過程である。

240 **秘匿性** - 秘匿性は、データが認可されていない個人、実体、またはプロセスからは利用できな
241 いという特性である。

242 **要約** - 要約は、オクテット・ストリームの暗号的チェックサムである。

243 **デジタル署名** - 本文書では、デジタル署名と署名は交換可能なように使われ、同じ意味をもつ。

244 **エンド・トゥ・エンドのメッセージ層セキュリティ** - エンド・トゥ・エンドのメッセージ層セ
245 キュリティは、ビジネス実体、例えば、企業、部門、ビジネス単位の内部や間の複数のアプリ
246 ケーション (1 つ以上の SOAP 仲介者) で受け渡されるメッセージが、これらのビジネス実体
247 を経ながらすべての経路に渡って安全であるときに確立される。それらが Web Services また
248 は伝統的なメッセージであるか否かに拘わらず、これには、実体内部で開始するメッセージだ
249 けでなく、実体外部から発生したメッセージも含まれる。

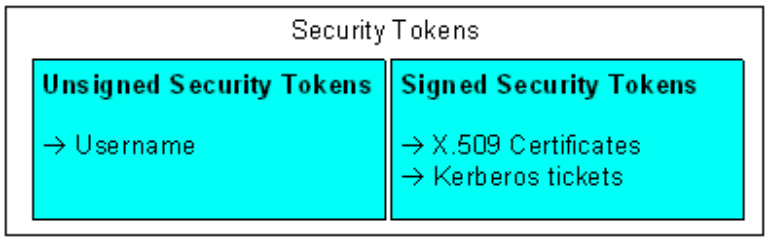
250 **完全性** - 完全性は、データが改変されなかったという特性である。

251 **メッセージ秘匿性** - メッセージ秘匿性はメッセージの特性であり、暗号化はメッセージのこの
252 特性を提供する機構である。

253 **メッセージ完全性** - メッセージ完全性はメッセージの特性であり、デジタル署名はメッセージ
254 のこの特性を提供する機構である。

署名 - 署名は、暗号的アルゴリズムで計算された値であり、その署名はデータの正式な受信者がデータ改竄のないことや、またメッセージの署名者から来たものであることを検証するために利用できるようにデータに結び付けられる。メッセージの完全性と認証を提供する。署名は、署名と検証に同じ鍵が利用される対称鍵アルゴリズムで計算し検証することができるし、または、異なる鍵が署名と検証に利用される (私有と公開鍵の組が利用される) 非対称鍵アルゴリズムを利用することもできる。

セキュリティ・トークン - セキュリティ・トークンは (1 つまたはそれ以上の) 申告の集まりを表現する。



署名付きセキュリティ・トークン - 署名付きセキュリティ・トークンは、特定の権威者によって主張され暗号的に署名されたセキュリティ・トークンである (例えば、X.509 証明書または Kerberos チケット)。

信頼 - 信頼は、ある実体が動作の集合を実行したり主体や範囲の集合についての表明の集合を作成することを別の実体に頼るという特徴である。

3メッセージ保護機構

SOAP メッセージを安全にするとき、様々な種類の脅威が考慮される必要がある。これには、次のことが含まれるが、これに限定されない。

メッセージが敵対者によって改変または読み取りされることができる。または、

敵対者が、処理を保証する適切なセキュリティ申告に欠けているが、適格であるメッセージをサービスに送付することができる。

これらの脅威を理解するために、本規定ではメッセージ・セキュリティ・モデルを定義する。

3.1 メッセージ・セキュリティ・モデル

本文書は、SOAP メッセージの保護と認証のために、デジタル署名と組み合わせたセキュリティ・トークンによって、抽象的なメッセージ・セキュリティ・モデルを規定する。

セキュリティ・トークンは申告を主張し、認証秘密または鍵とセキュリティ身元との間の束縛を主張するために利用されることができる。権威者は、セキュリティ・トークンに署名または暗号のために自身の鍵を利用することよって、セキュリティ・トークン中の申告を保証または裏書することができる。(鍵による暗号を利用することが推奨される) これにより、トークン中の申告の認証を可能とする。 1 つの身元と公開鍵の間の束縛を申告する X.509 証明書 [X509] は、証明書権威者によって裏書された署名付きセキュリティ・トークンの例である。第三者による裏書がなければ、セキュリティ・トークンの受信者は、含んでいるメッセージの製作者への信頼にもとづいてトークン中でなされる申告を受け入れるかどうかを選択してもよい。

メッセージの起源と完全性を検証するために署名が利用される。署名はまた、メッセージの製作者によって、セキュリティ・トークン中の申告を確認するために利用される、通常は第三者による、鍵に関する知識を実証するために利用される。こうして、それらの身元 (および、セキュリティ・トークン中に現れる他の申告) をそれらが作成したメッセージへ束縛する。

このセキュリティモデルは、それ自身、複数のセキュリティ攻撃を受けやすいことに注意すべきである。詳細については、Security Considerations の節を参照のこと。

規定で要素が「処理される」ことを必要とする場合、その要素型は、要素がサポートされない場合に適切なエラーを返すという程度まで認識されていなければ**ならない (MUST)**。

3.2 メッセージ保護

メッセージの内容を、開示されることから防ぐこと (秘匿性) または検知されることなく改変されることから防ぐこと (完全性) は、第一のセキュリティの関心事である。本規定では、ボディ、ヘッダ、またはその (またはその部分の) 任意の組合せを暗号化し、そして/または、デジタル的に署名することによってメッセージを守る方法を提供する。

メッセージ完全性は、メッセージへの改変が検知されることを請け負うために、セキュリティ・トークンと一緒に XML Signature [XMLSIG] によって提供される。完全性の機構は、もしかすると複数の SOAP actor/role による複数の署名をサポートし、追加の署名形式をサポートするために拡張できるように設計されている。

メッセージ完全性は、SOAP メッセージの部分を秘密に保持するために、セキュリティ・トークンと一緒に XML Encryption を利用する。暗号機構は、追加の暗号処理と複数の SOAP actor/role による操作をサポートするよう設計されている。

本文書では、<wsse:Security> 要素内の署名の構文とセマンティクスを定義する。

本文書は、<wsse:Security> 要素の外に現れるいかなる署名も規定しない。

3.3 無効または不足した申告

メッセージ受信者は、無効な署名をもつメッセージ、必要な申告が不足しているメッセージ、またはその申告が受理できない値をもつメッセージを拒否する**必要がある (SHOULD)**。そのようなメッセージは認可されない (または、奇形の)。本規定では、メッセージに 0 個以上のセキュリティ・トークンを関連付けることによって、セキュリティ特性に関する申告を行なうための柔軟な方法を、メッセージ製作者に提供する。セキュリティ申告の例は、製作者の身元である。製作者は、彼が Bob であり、ある会社の従業員として知られており、それゆえ彼はメッセージを送信する権利を持っていると申告することができる。

3.4 例

次の例はカスタム・セキュリティ・トークンと関連付けられた署名の利用を示している。トークンは、受信者により適切に認証されることができると我々が仮定する対称鍵を運ぶ、base64 符号化されたバイナリ・データを含む。メッセージ送信者はメッセージに署名するために HMAC 署名アルゴリズムを用いて対称鍵を利用する。メッセージ受信者は、HMAC 鍵計算を繰返すための共有鍵の知識を利用し、その鍵を署名が有効であると確認するために利用し、プロセスにおいて、メッセージが申告された利用者身元から作られたことを確認する。

```
(001) <?xml version="1.0" encoding="utf-8"?>
(002) <S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
            xmlns:ds="...">
(003)    <S11:Header>
(004)      <wsse:Security
             xmlns:wsse="...">
(005)        <xxx:CustomToken wsu:Id="MyID"
                              xmlns:xxx="http://fabrikam123/token">
(006)            FHUIORv...
(007)        </xxx:CustomToken>
(008)        <ds:Signature>
(009)           <ds:SignedInfo>
(010)              <ds:CanonicalizationMethod
                       Algorithm=
                         "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(011)              <ds:SignatureMethod
                       Algorithm=
                       "http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
(012)              <ds:Reference URI="#MsgBody">
(013)                 <ds:DigestMethod
                         Algorithm=
                       "http://www.w3.org/2000/09/xmldsig#sha1"/>
(014)                 <ds:DigestValue>LyLsF0Pi4wPU...</ds:DigestValue>
(015)              </ds:Reference>
(016)           </ds:SignedInfo>
(017)           <ds:SignatureValue>DJbchm5gK...</ds:SignatureValue>
(018)           <ds:KeyInfo>
(019)               <wsse:SecurityTokenReference>
```

```
(020)                    <wsse:Reference URI="#MyID"/>
(021)                </wsse:SecurityTokenReference>
(022)             </ds:KeyInfo>
(023)          </ds:Signature>
(024)       </wsse:Security>
(025)    </S11:Header>
(026)    <S11:Body wsu:Id="MsgBody">
(027)      <tru:StockSymbol xmlns:tru="http://fabrikam123.com/payloads">
               QQQ
           </tru:StockSymbol>
(028)    </S11:Body>
(029) </S11:Envelope>
```

最初の 2 行は SOAP エンベロープを開始する。 (003) 行は、この SOAP メッセージに関連付けられたヘッダを開始する。

(004) 行は、本規定で定義される <wsse:Security> ヘッダを開始する。このヘッダには、意図された受信者のためのセキュリティ情報を含む。この要素は (026) 行まで続く。

(005) 行から (007) は、メッセージに関連付けられたカスタム・トークンを指定する。この場合、外部で定義されたカスタム・トークン形式を利用している。

(008) 行から (023) はデジタル署名を指定する。この署名は署名された要素の完全性を請け負う。署名には、(002) 行での ds 名前空間宣言により識別される XML Signature 規定を利用する。

(009) 行から (016) は、何が署名されているのかと、カノニカリゼーションのどの型が利用されているかを記述する。 (010) 行は、署名されたデータをどのようにカノニカライズ (正規化) するかを指定する。 (012) 行から (015) は、署名された要素とそれらをどのようにダイジェストするかを選択する。特に、(012) 行は <S11:Body> 要素が署名されていることを示している。この例では、メッセージのボディのみが署名される。典型的には、メッセージ中のすべての重要な要素が署名に含まれる (以下の Extended Example を参照)。

(017) 行は XML Signature 規定で定義された、署名されたデータのカノニカライズされた形式の署名値を指定する。

(018) 行から (022) は、この署名に関連付けられたセキュリティ・トークンをどこで見つけるかの情報を、部分的または完全に、提供する。特に、(019) 行から (021) は、指定した URL (から引出して) で見つけられることができるセキュリティ・トークンを示している。

(026) 行から (028) は SOAP メッセージのボディ (ペイロード) を含む。

4ID 参照

署名参照のような他のメッセージ要素を参照するまたは署名をセキュリティ・トークンへ関連付けることに対して、多くの動機付けがある。このため、本規定は、受信者がセキュリティ要素の処理のためにメッセージの全スキーマを理解しなくてもよいように wsu:Id 属性を定義する。すなわち、彼らは、 wsu:Id 属性は要素を参照するために利用される ID のスキーマ型を表現するということを「知る」だけでよい。しかしながら、本規定で利用されるいくつかの主要なスキーマ(はっきり書くと、XML Signature と XML Encryption)は属性の拡張を許さないため、本規定ではまた wsu:Id 属性に加えてそれらのローカルの ID 属性の利用を許す。その結果として、署名の中で参照される要素の場所を見つけようとするとき、次の属性が考慮される。

XML Signature 要素上のローカル ID 属性

XML Encryption 要素上のローカル ID 属性

要素上の (以下で説明される) 大域 wsu:Id 属性

加えて、ボディのようなエンベロープの一部に署名するとき、より一般的な変換 (特に XPath [XPATH]) の代わりに ID 参照が利用されることが**推奨される (RECOMMENDED)**。これは処理を単純化するためである。

4.1 Id 属性

SOAP メッセージ内の要素が参照される必要のある多くの状況が存在する。例えば、SOAP メッセージに署名するとき、選択された要素が署名の範囲に含まれる。 XML Schema Part 2 [XMLSCHEMA] は要素を識別し参照するために利用されるかもしれないいくつかの組込みデータ型を提供するが、それらの利用には SOAP メッセージの消費者が、識別または参照の機構が定義されるスキーマを保持するか、取得可能でなければならないことが必要とされる。いくつかの環境では、例えば仲介者は、これは問題であり、望ましくない。

そのため、SOAP 基礎に基づいて、要素が利用される文脈の完全なスキーマ知識に頼らないような、要素を識別し参照するための機構が必要とされる。要素が動的なスキーマ発見と処理をすることなしに識別され参照されることができるように、この機能を SOAP 処理装置へ統合することができる。

本節では、任意の属性を許すまたは特定の属性を明確に許す要素へ適用されることができる要素を識別するための、名前空間限定の大域属性を定める。

4.2 Id スキーマ

仲介者と受信者にとって処理を簡単にするために、共通の属性が要素を識別するために定義される。この属性は XML Schema の ID 型を利用し、要素に対するこの情報を示すための共通の属性を指定する。

この属性の構文は次の通りである。

```
<anyElement wsu:Id="...">...</anyElement>
```

上で示した属性を次に示す。

.../@wsu:Id

型 xsd:ID として定義されるこの属性は、要素のローカルな ID を指定するためのよく知られた属性を提供する。

1 つの XML 文書中の 2 つの wsu:Id 属性は同じ値を持っては**ならない (MUST NOT)**。実装は、文書内唯一性を基本的に強制するために XML Schema 妥当性検証に頼っても**よい (MAY)**。しかしながら、アプリケーションは唯一性を強制するためにスキーマ妥当性検証のみに頼らない**ほうがよい (SHOULD NOT)**。

本規定は、この属性がどのように利用されるかを規定しないし、他の規定がこの属性の利用に対して付加的なセマンティクス (または制限) を加え**てもよい (MAY)** ことが期待される。

次の例は、要素を識別するためにこの属性の利用を示す。

```
<x:myElement wsu:Id="ID1" xmlns:x="..." 413
           xmlns:wsu="..."/>
```

XML Schema をサポートする準拠した処理装置は、この属性を、大域属性宣言を利用して定義されたかのように扱わなければ**ならない (MUST)**。

動的 XML Schema または DTDs の発見と処理をサポートしない準拠した処理装置は、この属性の定義をパーサに統合することを強く勧められる。すなわち、{target namespace} が "`http://www.w3.org/2001/XMLSchema`" で {name} が "Id" である [type definition] をその PSVI がもつかのようにこの属性情報項目を取り扱うことである。そうすることによって、処理装置は、関連したスキーマを見つけて処理する必要なしに属性を処理する方法を本質的に知ることができる。特に、実装は XML Signature 参照との相互運用のために XPointer [XPointer] shorthand pointer として利用のための有効な識別子として wsu:Id の値をサポートしても**よい (MAY)**。

5Security ヘッダ

<wsse:Security> ヘッダ部は SOAPactor/role の形式での特定の受信者を対象としたセキュリティ関連情報を付与するための機構を提供する。これは最終的なメッセージ受信者でも仲介者のどちらでもよい。したがって、この型の要素は SOAP メッセージに複数回存在してもよい。メッセージ経路上のアクティブな仲介者は、もしそれらがその SOAP ノードを対象としたものであれば、既存の <wsse:Security> ヘッダ部へ 1 つまたはそれ以上の新しい下位要素を付加しても**よい (MAY)** し、またはそれは追加の対象に対して 1 つまたはそれ以上の新しいヘッダを追加しても**よい (MAY)**。

先に述べたように、メッセージが別個の受信者を対象とするなら、複数の <wsse:Security> ヘッダ部をもってもよい (MAY)。しかしながら、1 つの <wsse:Security> ヘッダ部だけが S11:actor または S12:role 属性を省略してもよい (MAY)。 2 つの <wsse:Security> ヘッダ部が S11:actor または S12:role に対して同じ値を持っては**ならない (MUST NOT)** 。異なる受信者を目的とするメッセージ・セキュリティ情報は、異なる <wsse:Security> ヘッダ部に現れなくては**ならない (MUST)**。これは可能性のある処理順序の問題 (例えば、可能なヘッダの再順序付けによる) による。指定された S11:actor または S12:role のない <wsse:Security> ヘッダ部は、誰によって処理されても**よい (MAY)** が、最後の送り先または終点の前に削除されては**ならない (MUST NOT)**。

要素が <wsse:Security> ヘッダ部に追加されるとき、それらは既存の要素の前に挿入される**必要がある (SHOULD)**。したがって、<wsse:Security> ヘッダ部は、メッセージ製作者がメッセージを作成するためにとった署名と暗号化のステップを表している。前に挿入するというこの規則は、下位要素間での前方依存がないであろうため、受信するアプリケーションが <wsse:Security> ヘッダ部に現れる順序で下位要素を処理できることを保証する。本規定では下位要素の特定の処理順序を強いないことに注意すること。受信するアプリケーションは必要とされるどのような順序も利用することができる。

下位要素が他の下位要素で運ばれた鍵を参照するとき (例えば、署名に利用される X.509 証明書を含むバイナリ・セキュリティ・トークン下位要素を参照する署名下位要素)、鍵運搬要素が、その鍵を利用する要素より前に順序付けられる**必要がある (SHOULD)**。

```
<S11:Envelope>
    <S11:Header>
            ...
        <wsse:Security S11:actor="..." S11:mustUnderstand="...">
            ...
        </wsse:Security>
            ...
    </S11:Header>
    ...
</S11:Envelope>
```

上の例で列挙された属性と要素を次に示す。

/wsse:Security

受信者にセキュリティに関連するメッセージ情報を渡すためのヘッダ部である。

/wsse:Security/@S11:actor

483 この属性は識別されるべき特定の SOAP 1.1 [SOAP11] actor を指定する。この属性はオプ
484 ションである。しかしながら、ヘッダ部の 2 つのインスタンスが actor を省略したり、同じ
485 actor を指定することはできない。

486 /wsse:Security/@S12:role

487 この属性は識別されるべき特定の SOAP 1.2 [SOAP12] role を指定する。この属性はオプシ
488 ョンである。しかしながら、ヘッダ部の 2 つのインスタンスが role を省略したり、同じ role
489 を指定することはできない。

490 /wsse:Security/{any}

491 これは、スキーマに基づいて、渡されるべきセキュリティ情報の異なる (拡張可能な) 型を
492 許す拡張性機構である。認識されない要素は失敗を発生する**必要がある (SHOULD)**。

493 /wsse:Security/@{any}

494 これは、スキーマに基づいて、ヘッダに追加されるべき付加的な属性許す拡張性機構である。
495 認識されない属性は失敗を発生する**必要がある (SHOULD)**。

496 すべての準拠した実装は <wsse:Security> 要素を処理できなければならない (MUST)。

497 すべての準拠した実装は、それらがどのプロファイルをサポートするかを宣言**しなければなら**
498 **ず (MUST)**、また、そのプロファイルで定義されるかもしれない下位要素を含む
499 <wsse:Security> 要素を処理**できなければならない (MUST)**。 <wsse:Security> ヘッダ
500 内の定義されていない要素は処理されないことが**推奨される (RECOMMENDED)**。

501 次のいくつかの節で <wsse:Security> ヘッダ内で利用されると予期される要素の概要を説
502 明する。

503 <wsse: Security> ヘッダ が mustUnderstand="true" 属性を含む場合:

504 受信者は、名前空間に対応する WSS: SOAP Message Security 規定を実装しないなら SOAP
505 フォルトを生成しなければ**ならない (MUST)**。実装は、WSS: SOAP Message Security で規定
506 された必要とされる処理規則にしたがうと同時に、スキーマを解釈する能力を意味する。

507 受信者は、対応する WSS: SOAP Message Security トークン・プロファイルにしたがって、
508 <wsse:Security> ヘッダ部中に含まれるセキュリティ・トークンを解釈または処理できないな
509 らば、フォルトを生成しなければならない。

510 受信者は、ローカルなセキュリティ・ポリシーに基づいて、<wsse:Security> 要素内の要素や
511 拡張を無視しても**よい (MAY)**。

6セキュリティ・トークン

本章ではいくつかの異なる型のセキュリティ・トークンと、それらがどのようにメッセージに付加されるかを説明する。

6.1 セキュリティ・トークンの付与

本規定では、SOAP メッセージに関するセキュリティ情報を SOAP メッセージとともに伝えるための機構として <wsse:Security> ヘッダを定義する。このヘッダは、設計により、多くの型のセキュリティ情報をサポートするために拡張可能なように設計されている。

XML ベースのセキュリティ・トークンに対しては、<wsse:Security> ヘッダの拡張可能性により、ヘッダへこれらのセキュリティ・トークンを直接挿入することができる。

6.1.1 処理規則

本規定では XML Signature と XML Encryption を利用し処理するための処理規則を説明する。どの型のセキュリティ・トークンを利用するときもこれらの規則に従わなければ**ならない (MUST)**。署名または暗号化がセキュリティ・トークンと併せて利用されるときには、本規定により定義された処理規則に準拠した方法で利用されなければ**ならない (MUST)** ことに注意すること。

6.1.2 主体の確認

本規定では、申告の確認がされなければいけないか、どのようにされなければならないかは規定しない。しかしながら、署名がどのように利用され、申告確認の形式として(署名からセキュリティ・トークンを参照することによって)セキュリティ・トークンと関連付けられてもよいかを定義する。

6.2 利用者名トークン

6.2.1 利用者名

<wsse:UsernameToken> 要素が利用者名を提示する方法として導入される。この要素はオプションとして <wsse:Security> ヘッダに含められる。

次にこの要素の構文を示す。

```
<wsse:UsernameToken wsu:Id="...">
    <wsse:Username>...</wsse:Username>
</wsse:UsernameToken>
```

次に上の例でリストされた属性と要素を説明する。

/wsse:UsernameToken

　この要素は申告された身元を表現するために利用される。

/wsse:UsernameToken/@wsu:Id

このセキュリティ・トークンのための文字列ラベル。

/wsse:UsernameToken/wsse:Username

これは必須の要素で申告された身元を指定する。

/wsse:UsernameToken/wsse:Username/@{any}

これは、スキーマに基づいて、付加的な属性が <wsse:Username> 要素に追加されることを許す拡張機構である。

/wsse:UsernameToken/{any}

これは異なる (拡張可能な) 型のセキュリティ情報が渡されることを許すための拡張機構である。認識されない要素は失敗を発生する**必要がある (SHOULD)**。

/wsse:UsernameToken/@{any}

これは、スキーマに基づいて、追加の属性が <wsse:UsernameToken> 要素に与えられることを許す拡張機構である。認識されない属性は失敗を発生する**必要がある (SHOULD)**。

すべての準拠する実装は <wsse:UsernameToken> 要素を処理できなければ**ならない (MUST)**。次にこれの利用例を示す。

```
<S11:Envelope xmlns:S11="..." xmlns:wsse="...">
    <S11:Header>
            ...
        <wsse:Security>
            <wsse:UsernameToken>
                <wsse:Username>Zoe</wsse:Username>
            </wsse:UsernameToken>
        </wsse:Security>
            ...
    </S11:Header>
    ...
</S11:Envelope>
```

6.3 バイナリ・セキュリティ・トークン

6.3.1 セキュリティ・トークンの添付

バイナリ形式のセキュリティ・トークンに対して、本規定では <wsse:Security> ヘッダ部へ含めることができる <wsse:BinarySecurityToken> 要素を提供する。

6.3.2 バイナリ・セキュリティ・トークンの符号化

バイナリ・セキュリティ・トークン (例えば、X.509 証明書と Kerberos [KERBEROS] チケット) または他の非 XML 形式を使用するためには特別な符号化形式が必要となる。本節では、バイナリ・セキュリティ・トークンを利用するための基本的な枠組みを説明する。特定のバイナリ・セキュリティ・トークンの形式を作成し処理する規則はそれぞれ個別規定の中で定められなければ**ならない (MUST)**。

<wsse:BinarySecurityToken> 要素を解釈するために利用される 2 つの属性が定義される。ValueType 属性はそのセキュリティ・トークンが何か、例えば Kerberos チケット、を示す。

EncodingType はそのセキュリティ・トークンがどのように符号化されているか、例えば Base64Binary、を表す。次は構文の概要である。

```
<wsse:BinarySecurityToken wsu:Id=...
```

```
    EncodingType=...
    ValueType=.../>
```

次は上の例に出てくる属性と要素を説明する。

/wsse:BinarySecurityToken

この要素はバイナリに符号化されたセキュリティ・トークンを含めるために利用される。

/wsse:BinarySecurityToken/@wsu:Id

このセキュリティ・トークンのためのオプションの文字列ラベル。

/wsse:BinarySecurityToken/@ValueType

ValueType 属性は、符号化されたバイナリ・データの "値空間" を示すために利用される (例えば、X.509 証明書)。 ValueType 属性は、値の型と符号化されたバイナリ・データの空間を定義する URI を許す。各個別仕様では、それぞれが定義するトークンに対する ValueType 値を定義しなければ**ならない (MUST)**。 ValueType の使用が**推奨される (RECOMMENDED)**。

/wsse:BinarySecurityToken/@EncodingType

EncodingType 属性は、URI を利用して、バイナリ・データの符号化形式 (例えば base64 符号化) を示すために利用される。現在のスキーマ検証ツールでは、XML Schema 中で単純と複合型が混在している派生を作ることが困難であるという問題により、導入された新しい属性である。 EncodingType 属性は要素の符号化形式を示すために解釈される。次の符号化形式が事前定義されている (URI フラグメントは本規定の URI に相対的であることに注意)。

URI	説明
#Base64Binary (デフォルト)	XML Schema ベース 64 符号化

/wsse:BinarySecurityToken/@{any}

これは、スキーマに基いて、追加の属性が <wsse:Username> 要素に与えられることを許す拡張機構である。

すべての準拠する実装は <wsse:BinarySecurityToken> 要素を処理できなければ**ならない (MUST)**。

<wsse:BinarySecurityToken> が署名に含められるとき - すなわち、<ds:Signature> 要素から参照されるとき - カノニカリゼーション・アルゴリズム (例えば、Exclusive XML Canonicalization [EXC-C14N]) が属性または要素値の中で利用されている QName の名前空間接頭辞の認可されない置換を許さないように注意すべきである。特に、もしトークンが検証用の鍵を運ばない(そして、その結果それは暗号的に署名に結びつけられない)ならばこれらの名前空間接頭辞は <wsse:BinarySecurityToken> 要素の中で宣言しておくことが**推奨される (RECOMMENDED)**。

6.4 XML トークン

本節は XML ベースのセキュリティ・トークンの利用のための枠組を示す。特定の XML ベースのセキュリティ・トークン形式に対する規則と処理方法については、個々のプロファイル仕様で規定する。

6.4.1 セキュリティ・トークンの識別と参照

本規定はまた、wsu:Id 属性と <wsse:SecurityTokenReference> 要素 (いくつかの追加の機構と同様に) を利用してセキュリティ・トークンを識別し参照するための複数の機構を定義する。適切な参照機構については特定のプロファイル文書を参照すること。しかしながら、特定の拡張を、<wsse:SecurityTokenReference> 要素に対して行っても**よい (MAY)**。

7 トークン参照

本章ではセキュリティ・トークンを参照するための機構について議論し定義する。

7.1 SecurityTokenReference 要素

セキュリティ・トークンは一連の申告を運ぶ。これらの申告はどこか別のところに存在する場合があり、受信アプリケーションによって "引き出さ" れる必要がある。<wsse:SecurityTokenReference> 要素はセキュリティ・トークンを参照するための拡張可能な機構を提供する。

すべてのトークンが共通の参照パターンをサポートしているわけではないため、<wsse:SecurityTokenReference> 要素はセキュリティ・トークンを参照するための開かれた内容モデルを提供する。同様に、いくつかのトークン形式は閉じたスキーマをもち、独自の参照機構を定義する。開かれた内容モデルは、対応するトークン型を参照するときに、適切な参照機構が使われることを可能にする。

<wsse:SecurityTokenRefeference> が <wsse:Security> ヘッダ部の外で利用される場合、応答の意味および/または結果として起こる参照の処理規則は含まれている要素によって指定されなければ**ならず (MUST)**、本規定の範囲外である。

この要素の構文を以下に示す。

```
<wsse:SecurityTokenReference wsu:Id="...">
    ...
</wsse:SecurityTokenReference>
```

次に上で定義された要素を説明する。

/wsse:SecurityTokenReference

この要素はセキュリティ・トークンへの参照を示す。

/wsse:SecurityTokenReference/@wsu:Id

このセキュリティ・トークン参照に名前付けするための文字列ラベル。この属性は、参照されているものが何であるかの ID を示すのではなく、<wsse:SecurityTokenReference> 要素中の <wsse:Reference> 要素の中のフラグメント URI を利用して参照される**必要がある (SHOULD)**。

/wsse:SecurityTokenReference/@wsse:Usage

このオプションの属性は <wsse:SecurityToken> の使用法の分類のために利用される。使用法は URI を利用して指定され、XML リストのセマンティクスを利用して複数の使用法が指定されても**よい (MAY)**。本規定で定義される使用法はない。

/wsse:SecurityTokenReference/{any}

これは、スキーマに基づいて、異なる (拡張可能な) セキュリティ参照の型が渡されることを許す拡張機構である。認識されない要素は失敗を発生する**必要がある (SHOULD)**。

/wsse:SecurityTokenReference/@{any}

これは、スキーマに基づいて、追加の属性がヘッダに追加されることを許す拡張機構である。認識されない属性は失敗を発生する**必要がある (SHOULD)**。

すべての準拠する実装は <wsse:SecurityTokenReference> 要素を処理できなければ**ならない (MUST)**。

この要素はまた、どこか別の場所にあるセキュリティ・トークンから鍵情報を取得するためのヒントを示すために<ds:KeyInfo> の直接の子要素として利用してもよい。特に、XML Signature と XML Encryption を利用するとき、署名または暗号のために利用されるセキュリティ・トークンを参照するために<wsse:SecurityTokenReference> 要素を <ds:KeyInfo> の内部に置くことが**推奨される (RECOMMENDED)**。

相互運用を試そうとするときに、実装が直面するいくつかの難局がある。 ID と参照の処理は、受信者がスキーマを*理解する*ことを必要とする。このような実装は高価であり、特定の名前空間 URI に対する "スキーマの位置" を知る方法がないため、一般的な場合には不可能かもしれない。その上、参照の第一の目的は、必要なトークンを一意に識別することである。 ID 参照は、定義上は、XML によって一意である。しかしながら、"主体の名前" のような他の機構では一意である必要がなく、したがってそのような参照は一意でないかもしれない。

次のリストは WSS: SOAP Message Security で定義される限定的な参照機構のリストを優先順(すなわち、もっとも限定的なものからもっとも限定的でないものの順)に提供する:

直接参照 – これは URI フラグメントを利用した内部トークンへの参照と、完全 URI を利用した外部トークンへの参照を許す。

鍵識別子 – これは、(トークンの型/プロファイルにより定義された) トークンを表現する不透明な値を利用してトークンを参照することを許す。

鍵名 – これはトークンが、セキュリティ・トークン内のアイデンティティ表明に一致する文字列を利用して参照されることを許す。これは部分一致であり、指定された名前に一致する複数のセキュリティ・トークンが結果として得られるかもしれない。

埋め込まれた参照 – これは、(どこかに存在するトークンへのポインタではなくて)トークンが埋め込まれることを許す。

7.2 直接参照

<wsse:Reference> 要素は、URI を利用してセキュリティ・トークンを直接参照するための拡張可能機構を提供する。

この要素の構文を以下に示す。

```
<wsse:SecurityTokenReference wsu:Id="...">
    <wsse:Reference URI="..." ValueType="..."/>
</wsse:SecurityTokenReference>
```

上で定義された要素を以下に説明する。

/wsse:SecurityTokenReference/wsse:Reference

この要素は、セキュリティ・トークンの位置を見つけるための抽象的な URI 位置を識別するために利用される。

/wsse:SecurityTokenReference/wsse:Reference/@URI

このオプションの属性は、セキュリティ・トークンを見つける場所の抽象 URI を指定する。もしフラグメントが指定されたなら、参照されるトークンのローカルな ID が参照されていることを示す。

/wsse:SecurityTokenReference/wsse:Reference/@ValueType

このオプションの属性は、参照されているトークンの型を識別するために利用される URI を指定する。本規定ではこの属性に関する処理規則は定義しないが、個々のトークン型のための規定は、 URI の値に関する特定の処理規則とセマンティクス、および、それがどのように解釈される**ことになるか (SHALL)** を定義しても**よい (MAY)**。この属性が存在しないなら、URI は通常の URI として処理されなければ**ならない (MUST)**。ローカル URI での参照に対しては ValueType の利用が**推奨される (RECOMMENDED)**。

/wsse:SecurityTokenReference/wsse:Reference/{any}

これは、スキーマに基づいて、異なる (拡張可能な) セキュリティ参照の型が渡されることを許す拡張機構である。認識されない要素は失敗を発生する**必要がある (SHOULD)**。

/wsse:SecurityTokenReference/wsse:Reference/@{any}

これは、スキーマに基づいて、追加の属性がヘッダに追加されることを許す拡張機構である。認識されない属性は失敗を発生する**必要がある (SHOULD)**。

この要素の利用例を以下に示す。

```
<wsse:SecurityTokenReference
          xmlns:wsse="...">
    <wsse:Reference
              URI="http://www.fabrikam123.com/tokens/Zoe"/>
</wsse:SecurityTokenReference>
```

7.3 鍵識別子

別の方法として、直接参照が利用されない場合には、<ds:KeyName> の代わりにセキュリティ・トークンを指定/参照するために鍵識別子を利用することが**推奨される (RECOMMENDED)**。 KeyIdentifier はセキュリティ・トークンを一意に識別するために利用されることができる値である (例えば、セキュリティ・トークンの重要な要素のハッシュ値など)。正確な値の型と生成アルゴリズムはセキュリティ・トークン型によって (また時には、トークン中のデータによっても) 異なり、その結果、値とアルゴリズムは本規定ではなくトークン固有のプロファイルで記述される。

識別子を利用してトークンを参照するために、<wsse:KeyIdentifier> 要素が <wsse:SecurityTokenReference> 要素中に置かれる**ことになる (SHALL)**。この要素はすべての鍵識別子に対して利用される**必要がある (SHOULD)**。

処理モデルは、セキュリティ・トークンに対する鍵識別子が不変のものであることを仮定する。結果として、鍵識別子の処理は単純に、指定された定数に一致する鍵識別子をもつセキュリティ・トークンを単に探す事になる。

構文の概要を以下に示す。

```
<wsse:SecurityTokenReference>
    <wsse:KeyIdentifier wsu:Id="..."
                        ValueType="..."
                        EncodingType="...">
        ...
    </wsse:KeyIdentifier>
</wsse:SecurityTokenReference>
```

744 上の例でリストされた属性と要素を以下に説明する。

745 /wsse:SecurityTokenReference/wsse:KeyIdentifier

746 この要素はバイナリ符号化された鍵識別子を含めるために利用される。

747 /wsse:SecurityTokenReference/wsse:KeyIdentifier/@wsu:Id

748 この識別子のためのオプションの文字列ラベル。

749 /wsse:SecurityTokenReference/wsse:KeyIdentifier/@ValueType

750 オプションの ValueType 属性は、利用されている KeyIdentifier の型を示すために利用され
751 る。各個別のトークンのプロファイルは KeyIdentifier 型を規定し、その型のトークンを参照
752 するために利用してもよい。またそれは、 KeyIdentifier が鍵またはトークンに一意かどうか
753 というような、識別子の重要なセマンティクスも指定する。値が指定されないなら、鍵識別
754 子はアプリケーション固有の方法で解釈されるだろう。

755 /wsse:SecurityTokenReference/wsse:KeyIdentifier/@EncodingType

756 オプションの EncodingType 属性は、URI を利用して、KeyIdentifier の符号化形式(#
757 Base64Binary)を示すために利用される。本規定により定義される基本の値が利用される
758 (URI フラグメントは本文書の URI に相対的であることに注意):

759

URI	説明
#Base64Binary	XML Schema base 64 符号化(デフォルト)

760

761 /wsse:SecurityTokenReference/wsse:KeyIdentifier/@{any}

762 これは、スキーマに基づいて、追加の属性が追加されることを許す拡張機構である。

763 7.4 埋め込まれた参照

764 いくつかの場合、参照は(どこかに存在するトークンへのポインタではなくて)埋め込まれたト
765 ークンを示すものであってもよい。これを実現するために、<wsse:Embedded> 要素が
766 <wsse:SecurityTokenReference> 要素の中で指定される。

767 次が構文の概要である。

```
768 <wsse:SecurityTokenReference>
769    <wsse:Embedded wsu:Id="...">
770       ...
771    </wsse:Embedded>
772 </wsse:SecurityTokenReference>
```

773 上の例に示した属性と要素を以下に説明する。

774 /wsse:SecurityTokenReference/wsse:Embedded

775 この要素はトークンを参照中に直接埋め込むため(すなわち、ローカルのまたは文字による
776 参照を構成するため)に利用される。

777 /wsse:SecurityTokenReference/wsse:Embedded/@wsu:Id

778 この要素のためのオプションの文字列ラベル。これは埋め込まれたトークンが署名または暗
779 号によって参照されることを許す。

780 /wsse:SecurityTokenReference/wsse:Embedded/{any}

これは、スキーマに基づいて、任意のセキュリティ・トークンが埋め込まれることを許す拡張機構である。要素が認識されないときは失敗を発生する**必要がある (SHOULD)**。

/wsse:SecurityTokenReference/wsse:Embedded/@{any}

これは、スキーマに基づいて、追加の属性が追加されることを許す拡張機構である。認識されない属性は失敗を発生する**必要がある (SHOULD)**。

SAML 表明の埋め込みの例を以下に示す。

```
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="...">
    <S11:Header>
        <wsse:Security>
            ...
            <wsse:SecurityTokenReference>
                <wsse:Embedded wsu:Id="tok1">
                    <saml:Assertion xmlns:saml="...">
                        ...
                    </saml:Assertion>
                </wsse:Embedded>
            </wsse:SecurityTokenReference>
            ...
        <wsse:Security>
    </S11:Header>
    ...
</S11:Envelope>
```

7.5 ds:KeyInfo

(XML Signature の)<ds:KeyInfo> は鍵情報を運ぶために利用されることができ、異なる鍵の型と将来の拡張性のために許されている。しかしながら、本規定では、鍵の型がバイナリ・データを含むものであれば、 <wsse:BinarySecurityToken> の利用が鍵資材を運ぶための**推奨される (RECOMMENDED)** 機構である。鍵資材を運ぶための適切な方法については、特定のプロファイル文書を参照すること。

以下は、名付けられた鍵を取ってくための利用例を示している。

```
<ds:KeyInfo Id="..." xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:KeyName>CN=Hiroshi Maruyama, C=JP</ds:KeyName>
</ds:KeyInfo>
```

7.6 鍵名

<wsse:KeyIdentifier> 要素を利用することが強く**推奨される (RECOMMENDED)**。しかしながら、鍵の名前が利用されるならば、相互運用性のために <ds:KeyName> 要素が RFC 2253 (これは <X509SubjectName> に対して XML Signature によって推奨されている) の 2.3 節の属性名に準拠することが強く**推奨される (RECOMMENDED)**。

さらに、e-mail アドレスは RFC 822 に準拠する**必要がある (SHOULD)**。

```
    EmailAddress=ckaler@microsoft.com
```

820 # 8署名

821 メッセージ作成者は、メッセージが送信中に改変されたかどうかを決定し、特定のセキュリテ
822 ィ・トークンの中の申告がメッセージの製作者に適用されることを確認することを、メッセー
823 ジ受信者に可能にしたいかもしれない。

824 トークン鍵の申告に関連した確認鍵を知っていることを実証することは、付随するトークンの
825 申告を確認する。確認鍵を知っていることは、例えば、XML Signature を作成するためにその
826 鍵を利用することによって実証されるかもしれない。信頼する者の申告の受諾は、トークン中
827 のその秘密に依存するかもしれない。複数のトークンが署名に対する鍵の申告を含むかもしれ
828 ないし、<wsse:SecurityTokenReference> を利用して署名から参照されるかもしれない。鍵の
829 申告は、2 つ例を挙げると、 X.509 証明書トークンまたは Kerberos サービスのチケットトー
830 クンであるかもしれない。

831 いくつかの SOAP ヘッダの変わりやすさのため、製作者は XML Signature で定義される
832 Enveloped Signature Transform を利用**しないほうがよい (SHOULD NOT)**。その代わりに、メ
833 ッセージは、署名される要素を明示的に含める**必要がある (SHOULD)**。同様に、製作者は
834 XML Signature [XMLSIG] で定義される Enveloping Signature を利用**しないほうがよい**
835 **(SHOULD NOT)**。

836 本規定では、メッセージへ付与されるべき複数の署名と署名形式を許す。それぞれ (の署名)
837 はメッセージの異なる、重なりさえある、部分を参照している。これは、メッセージが複数の
838 処理段階を通って流れるような多くの分散されたアプリケーションにとって重要である。例え
839 ば、製作者は orderID ヘッダを含んだ注文書を提出するかもしれない。製作者は orderID ヘッ
840 ダおよび要求の本体 (注文書の内容) に署名する。これが注文書処理下位システムにより受け
841 取られたとき、それはヘッダに shippingID を挿入するかもしれない。注文書下位システムは
842 それから、少なくとも orderID と shippingID に、そしておそらく本体も同様に署名するだろう。
843 それから、この注文書が処理され出荷部門により出荷されたとき、shippedInfo ヘッダが追加
844 されるかもしれない。出荷部門は少なくとも shippedInfo と shippingID に、そしておそらく本
845 体も同様に署名し、処理のためにメッセージを会計部門に回送するだろう。会計部門は署名を
846 検証し、注文書のための妥当な信頼の鎖を決定することができる。仮定の各ステップでだれが
847 是認したかも同様に。

848 すべての準拠した実装は XML Signature 標準に対応することができなければ**ならない (MUST)**。

849 8.1 アルゴリズム

850 本規定は XML Signature を基礎としており、そのため、XML Signature 規定で規定されたと同
851 じアルゴリズムへの要求をもつ。

852 次の表が本規定により強く**推奨される（RECOMMENDED）**付加的なアルゴリズムを概観する。

853

アルゴリズムの型	アルゴリズム	アルゴリズムの URI
Canonicalization	Exclusive XML Canonicalization	http://www.w3.org/2001/10/xml-exc-c14n#

854

855 同様に、次の表が利用されても**よい (MAY)** 追加のアルゴリズムを概観する:

アルゴリズムの型	アルゴリズム	アルゴリズムの URI
Transform	SOAP Message Normalization	http://www.w3.org/TR/2003/NOTE-soap12-n11n-20030328/

856

857 Exclusive XML Canonicalization アルゴリズムは一般的な正規化 (canonicalization) の落し穴を
858 扱う。既存の署名にある漏れやすい名前空間から起こる。

859 最後に、製作者が暗号化の前にメッセージに署名を望むなら、5 章 "Security Header" で書か
860 れた順序付け規則にしたがって、最初に <wsse:Security> ヘッダへ署名要素を前に加え、それ
861 から暗号要素を前に加える**必要があり (SHOULD)**、それによって最初に暗号要素、続いて署名
862 要素をもつ <wsse:Security> ヘッダを得ることにな。

863

<wsse:Security> header
[encryption element] [signature element] . .

864

865 同様に、製作者が暗号化の後にメッセージに署名を望むなら、最初に <wsse:Security> ヘッダ
866 へ暗号要素を前に加え、それから署名要素を前に加える**必要があり (SHOULD)**、それによって
867 最初に署名要素、続いて暗号要素をもつ <wsse:Security> ヘッダを得ることにな。

868

<wsse:Security> header
[signature element] [encryption element] . .

869

870 XML Digital Signature WG は 2 つの正規化アルゴリズムを定義した: XML Canonicalization と
871 Exclusive XML Canonicalization。混同を避けるため、最初のものは Inclusive Canonicailzation
872 とも呼ばれる。どちらも、起こり得るすべての問題を解決するわけではない。次の正式ではな
873 い議論は、ある状況でどちらを利用するかという選択のための手引きを提供することを意図し
874 ている。これらの問題について、より詳細で技術的に正確な議論は次を参照のこと: [XML-
875 C14N] と [EXC-C14N]。

876 避けられるべき 2 つの問題がある。一方では、XML は文書が様々な方法で変更されることを
877 許しながら、それでもなお等価と考えられる。例えば、二重の名前空間宣言は削除されること

878 もできるし、生成されることもできる。その結果、XML を処理するときに、XML ツールはこ
879 のような変更を自由に行なう。したがって、これら等価な形式が同じ署名に適合することが重
880 要である。

881 他方では、署名が単に xx:foo のようなものを覆うなら、xx が再定義されたとき、その意味が
882 変わるかもしれない。この場合、署名は改竄を防がない。この問題は、すべての値を整列して
883 展開することによって解決されると思われるかもしれない。不幸にも、xx と yy の両方が同じ
884 名前空間に結び付けられていたとしても、xx="http://example.com/"; が
885 yy="http://example.com/"; と異なると考えられる XPATH のような機構がある。

886 Inclusive と Exclusive Canonicalization の間の基本的な違いは、出力中に置かれる名前空間宣
887 言である。 Invlusive Canonicalization は、署名の範囲の外で定義されていたとしても、現在有
888 効ななすべての宣言を複製する。それはまた、xml:lang や xml:base のような、有効な xml: 属
889 性を複製する。これは、あなたが利用するかもしれないすべての宣言が曖昧なく指定されるこ
890 とを保証する。これに関する問題は、署名された XML が、他の宣言をもつ XML 文書中に移
891 されるなら、Inclusive Canonicalization は複製し、それから署名は無効となるであろう。これ
892 あ、あなたが単に囲っている文脈へ異なる名前空間の属性を追加するだけでも起こり得る。

893 Exclusive Canonicalization は、あなたがどの名前空間を実際に利用しているかを見つけ出そう
894 とし、それを複製するだけである。具体的に言うと、それは「目に見えて利用されている」も
895 のを複製する。それは XML 構文の一部であるようなものを意味する。しかしながら、それは
896 属性値や要素内容まで見ることはなく、これらを処理するために必要とされる名前空間宣言は
897 複製されない。例えば、xx:foo="yy:bar" のような属性があるとすると、xx に対する宣言は複
898 製するだろうが、yy に対する宣言は複製しないだろう。 (これは、もしあなたがスキーマ・サ
899 ブタイプを利用するなら、XML 処理ツールは xsi:type を追加するので、あなたの知らないと
900 ころで起こり得る。) それは、署名の範囲の外で宣言された xml: 属性も複製しない。

901 Exclusive Canonicalization は、あなたに、宣言されなければならない名前空間のリストを作る
902 ことを許す。それにより、目に見えて利用されていないものに対する宣言を拾い上げる。唯一
903 の問題は、署名を行なうソフトウェアが、それらが何なのかを知らなければならないというこ
904 とである。典型的な SOAP ソフトウェア環境では、セキュリティ・コードは、署名しようと
905 しているメッセージ本体にアプリケーションにより利用されているすべての名前空間を通常は
906 知らない。

907 Exclusive Canonicalization は、あなたが署名付き XML 文書を他の XML 文書へ挿入したいと
908 きに、便利である。良い例は、様々な SOAP メッセージのセキュリティ・ヘッダ中に XML
909 Token として挿入される、署名付き SAML アサーションである。アサーションに署名する
910 Issuer は、利用されている名前空間を知っていて、リストを構築できる。 Exclusive
911 Canonicalization の利用は、いつでも署名が正確に検証されることを保証する。

912 Inclusive Canonicalization は、本規定にしたがって SOAP 本体の部分またはすべてに署名する
913 という典型的な場合に便利である。これは、すべての専念が署名の下に落ちることを保証する。
914 どの名前空間が利用されているかをコードが知らなくても。同時に、署名されたデータ (及び、
915 署名要素) は何らかの他の XML 文書に挿入されることは少ないだろう。これが望まれたとし
916 ても、他の理由で実現可能でないかもしれない。例えば、両方の XML 文書に定義された同じ
917 値の Id があるかもしれない。

918 他の状況では、どちらが適切かを決定するためには、アプリケーションの要求と、正規化手段
919 の詳しい操作の調査が必要であろう。

920 本節は参考である。

8.2 メッセージへの署名

<wsse:Security> ヘッダ部は、1 つまたはそれ以上の要素に署名することを目的として SOAP Envelope 内に XML Signature 規定に準拠した署名を運ぶために利用されても**よい (MAY)**。複数の署名エントリが、単一の SOAP Envelope へ、<wsse:Security> ヘッダ部内に追加されても**よい (MAY)**。製作者はメッセージのすべての重要な要素に署名する**必要があり(SHOULD)**、送信中に合法的に改変されるかもしれないメッセージの部分に署名を要求する署名ポリシーを作成することに注意深い考慮がなされなければならない。

SOAP アプリケーションは次の条件を満さなければ**ならない (MUST)**。

準拠する実装は XML Signature 規定で定義された必要とされる要素を処理する能力がなければ**ならない (MUST)**。

<wsse:Security> ヘッダ部に署名を追加するためには、受信者に操作の正しい順序を示すよう、XML Signature 規定に準拠した <ds:Signature> 要素が <wsse:Security> ヘッダ部の既存の内容の前に追加されなければ**ならない (MUST)**。署名に含まれたすべての <ds:Reference> 要素は、XML Signature 仕様で説明されるように、取り囲んでいる SOAP エンベロープ内の資源を参照する**必要がある (SHOULD)**。しかしながら、SOAP メッセージ交換モデルは仲介アプリケーションが Envelope を変更する (例えば、ヘッダ部を追加または削除する) ことを許しているため、XPath フィルタリングの結果がメッセージ配送後も常に同じオブジェクトになるとは限らない。 XPath フィルタリングの利用には、そのような変更のためのその後の妥当性検証失敗がないよう、注意がとられる必要がある。

仲介者 (特にアクティブなもの) による変更の問題は XPath 処理以上にも適用できる。正規化 (canonicalization) とダイジェストのため、デジタル署名はそのような関係の特に脆い例を見せる。メッセージ処理全体が頑強のままであれば、仲介者は利用された変換アルゴリズムがデジタル的に署名された構成部品の妥当性に影響しないように注意しなければならない。

名前空間のセキュリティ懸念のため、本規定では "Exclusive XML Canonicalization" アルゴリズムまたはそれと同等かそれ以上の保護を提供する他の正規化 (canonicalization) アルゴリズムの利用を強く**推奨する (RECOMMENDS)**。

処理の効率のため、プロセサがトークンを利用する前に読み込みキャッシュできるように、署名が追加されてからセキュリティ・トークンが前に追加されることを**推奨する (RECOMMENDED)**。

8.3 トークンへの署名

メッセージの中またはメッセージのまさに外部に含まれたセキュリティ・トークンに署名することがしばしば望ましい。 XML Signature 規定は URI、ID、および XPath のように署名されるべき情報を参照するためのいくつかの一般的な方法を提供しているが、あるトークンは URI または ID を用いて参照されることを許さないかもしれないし、XPath はある状況では望ましくないかもしれない。

本規定では、異なるトークンが、<wsse:SecurityTokenReference> 要素への拡張として、それらのプロファイル中で指定されたそれらの一意の参照機構を持つことを許す。この要素は、すべてのトークン形式と連携することを保証された統一参照機構を提供する。したがって、本規定は XML Signature のための新しい参照オプションを定義する。それは STR Dereference Transform である。

この変換は URI #STR-Transform (URI フラグメントは本文書の URI に相対的なものであることに注意) により指定され、<wsse:SecurityTokenReference> 要素に適用される。それは出力が、要素自身ではなくて <wsse:SecurityTokenReference> 要素により参照されるトークンであることを意味する。

概要として、処理モデルは、<wsse:SecurityTokenReference> 要素に遭遇したときを除いて入力を変換にエコーすることである。 1 つが見つかるとき、その要素はエコーされないが、その代わり <wsse:SecurityTokenReference> 要素により定義された基準と規則に適合しているトークンの場所を見つけるために用いられ、それ (それら) を出力へエコーする。それ故に、変換の出力は、<wsse:SecurityTokenReference> 要素がマッチした参照されたセキュリティ・トークンに置換された入力を表現する結果としておこる系列である。

メッセージエンベロープ内に含まれるトークンを参照するこの変換の例を次に示す。

```
...
<wsse:SecurityTokenReference wsu:Id="Str1">
...
</wsse:SecurityTokenReference>
...
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
     <ds:SignedInfo>
      ...
       <ds:Reference URI="#Str1">
         <ds:Transforms>
           <ds:Transform
                Algorithm="...#STR-Transform">
             <wsse:TransformationParameters>
               <ds:CanonicalizationMethod
                      Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315" />
             </wsse:TransformationParameters>
           </ds:Transform>
           <ds:DigestMethod Algorithm=
                           "http://www.w3.org/2000/09/xmldsig#sha1"/>
           <ds:DigestValue>...</ds:DigestValue>
       </ds:Reference>
      </ds:SignedInfo>
     <ds:SignatureValue></ds:SignatureValue>
</ds:Signature>
...
```

次に上の例でリストされる属性と要素を説明する。

/wsse:TransformationParameters

この要素はトランスフォーメイションのためのパラメタをラップするために利用される。 XML Signature 名前空間からのものも。

/wsse:TransformationParameters/ds:Canonicalization

これは、選択されたデータに適用されるカノニカリゼーション・アルゴリズムを指定する。

/wsse:TransformationParameters/{any}

これは、将来的に、異なる (拡張可能な) パラメタが指定されることを許す拡張機構である。 認識されないパラメタは失敗を発生する**必要がある (SHOULD)**。

/wsse:TransformationParameters/@{any}

これは、将来的にスキーマに基づいて、追加の属性が要素に追加されることを許す拡張機構である。認識されない属性は失敗を発生する**必要がある (SHOULD)**。

次のものが変換の詳細な仕様である。

アルゴリズムは URI: #STR-Transform によって識別される。

変換の入力:

- 入力はノード集合である。入力がオクテット列ならば、それは自動的に構文会席される。 XML Digital Signature [XMLSIG] を参照。

変換の出力:

- 出力はオクテット列である。

構文:

- 変換は単一の必須のパラメタ <ds:CanonicalizationMethod> をもち、入力ノード集合のシリアライズのために利用される。しかしながら、出力は厳密にはカノニカリゼーション・アルゴリズムによるカノニカル形式でないかもしれない。しかしならが、出力は曖昧ではないという意味でカノニカルである。しかしながら、XML Signature 定義における構文要求のため、このパラメタは <wsse:TransformationParameters> 要素に覆われなければ**ならない (MUST)**。

処理規則:

- N を入力ノード集合とする。
- R を N の中のすべての <wsse:SecurityTokenReference> 要素の集合とする。
- R 中の各 Ri に対して、Di を Ri を dereference した結果とする。
- Di が決定されることができなければ、変換は失敗を発しなければ**ならない (MUST)**。
- もし Di が XML セキュリティ・トークン (例えば、SAML アサーションや <wsse:BinarySecurityToken> 要素) であれば、Ri' を Di とする。そうでなければ、Di は生のバイナリ・セキュリティ・トークンである。すなわち、オクテット列である。この場合、Ri' を <wsse:BinarySecurityToken> 要素から構成されるノード集合とする。<wsse:SecurityTokenReference> 要素 Ri と同じ名前空間前置詞を利用した。EncodingType 属性をもたない。セキュリティ・トークンの内容を識別する ValueType 属性、およびバイナリに符号化されたセキュリティ・トークンから構成されるテキスト内容。ホワイトスペースなしで。
- 最後に、この変換のオクテット・ストリーム出力をせいせいするために、N をシリアライズするために変換へのパラメタとして指定されていたカノニカリゼーション手法を採用する。しかし、dereference された <wsse:SecurityTokenReference> 要素 Ri とその子孫の場所には、その代わりに dereferenced ノード集合 Ri' を処理する。このステップの間、置換ノード集合のカノニカリゼーションは次のように増大されなければ**ならない (MUST)**。
  - 注意: 名前空間宣言 xmlns="" は、デフォルトの名前空間に対する値を宣言する名前空間ノードなしの各 apex 要素と共に発せられなければ**ならない (MUST)**。 XML Decryption Transform を参照。

8.4 署名妥当性検証

<wsse:Security> ヘッダ部内の <ds:Signature> 要素の妥当性検証は次のいずれかの場合に失敗することになる **(SHALL)**。

- 要素の内容の構文が本規定に準拠していない。
- XML Signature 規定 [XMLSIG] の核心の妥当性検証によって、要素に含まれる署名の妥当性検証に失敗した。
- それ自身の妥当性検証ポリシーを適用するアプリケーションが、何らかの利用でメッセージを拒否した (例えば、署名が信頼されない鍵により作成された - 前の 2 つのステップは署名の暗号法的な妥当性検証を達成するだけである)

署名要素の妥当性検証に失敗するなら、アプリケーションは「12 エラー処理」の節で定義された誤り符号を利用して製作者に失敗を報告してもよい **(MAY)**。

8.5 例

次の見本メッセージは、完全性とセキュリティ・トークンの利用を示す。この例では、メッセージ本体のみが署名される。

```
<?xml version="1.0" encoding="utf-8"?>
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
xmlns:ds="...">
   <S11:Header>
      <wsse:Security>
         <wsse:BinarySecurityToken
                    ValueType="...#X509v3"
                    EncodingType="...#Base64Binary"
                    wsu:Id="X509Token">
                 MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh5i...
         </wsse:BinarySecurityToken>
         <ds:Signature>
            <ds:SignedInfo>
               <ds:CanonicalizationMethod Algorithm=
                     "http://www.w3.org/2001/10/xml-exc-c14n#"/>
               <ds:SignatureMethod Algorithm=
                     "http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
               <ds:Reference URI="#myBody">
                  <ds:Transforms>
                     <ds:Transform Algorithm=
                           "http://www.w3.org/2001/10/xml-exc-c14n#"/>
                  </ds:Transforms>
                  <ds:DigestMethod Algorithm=
                        "http://www.w3.org/2000/09/xmldsig#sha1"/>
                  <ds:DigestValue>EULddytSo1...</ds:DigestValue>
               </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>
              BL8jdfToEb1l/vXcMZNNjPOV...
            </ds:SignatureValue>
            <ds:KeyInfo>
                <wsse:SecurityTokenReference>
                    <wsse:Reference URI="#X509Token"/>
                </wsse:SecurityTokenReference>
            </ds:KeyInfo>
         </ds:Signature>
```

```
      </wsse:Security>
   </S11:Header>
   <S11:Body wsu:Id="myBody">
      <tru:StockSymbol xmlns:tru="http://www.fabrikam123.com/payloads">
        QQQ
      </tru:StockSymbol>
   </S11:Body>
</S11:Envelope>
```

9暗号化

本規定では、製作者および受信者で共有された共通の対称鍵、または、メッセージ中に暗号化された形式で運ばれる対称鍵による、ボディ部、ヘッダ部と、その下位構造を任意の組合せで暗号化することを許す。

この柔軟性を許すために、本規定では XML Encryption 標準を利用する。具体的に言うと、本規定で記述するものは、<wsse:Security> ヘッダ部の中で (以下に列挙され、また、 XML Encryption で定義される) 3 つの要素がどのように利用されることができるかである。作成者またはアクティブな仲介者が XML Encryption を利用して SOAP メッセージの一部分を暗号化するとき、下位要素を <wsse:Security> ヘッダ部の前へ追加しなければ**ならない (MUST)**。さらに、暗号化する者は下位要素を、意図した受信者のための既存の <wsse:Security> ヘッダ部へ前に追加、または、新しい <wsse:Security> ヘッダ部を作り下位要素を挿入しなければ**ならない (MUST)**。これ以降、メッセージの一部分の暗号化と、これらの下位要素の 1 つを追加することの組合せ処理は、暗号化ステップと呼ばれる。下位要素は、受信者が復号できるメッセージの部分を識別するために、受信者に対して必要な情報を含まなければ**ならない (MUST)**。

すべての準拠した実装は XML Encryption 標準 [XMLENC] に対応することができなければ**ならない (MUST)**。

9.1 xenc:ReferenceList

XML Encryption [XMLENC]の <xenc:ReferenceList> 要素が、エンベロープ内の <xenc:EncryptedData> 要素として表現される暗号化された部分の目録を作るために使われても**よい (MAY)**。この暗号化ステップによって暗号化される要素または要素内容は、XML Encryption にしたがって、対応する <xenc:EncryptedData> 要素によって置換されなければ**ならない (MUST)**。この暗号化ステップで作成されるすべての <xenc:EncryptedData> 要素は、<xenc:ReferenceList> 要素の中の 1 つまたはそれ以上の <xenc:DataReference> 要素に列挙される**必要がある (SHOULD)**。

XML Encryption [XMLENC] の中で<xenc:ReferenceList> は、本来 <xenc:EncryptedKey> 要素の中で利用される (このことは、すべての参照される <xenc:EncryptedData> 要素が同じ鍵で暗号化されることを意味する) ために設計されたが、本規定では同じ <xenc:ReferenceList> により参照される <xenc:EncryptedData> 要素が異なる鍵により暗号化されても**よい (MAY)** ことを許す。それぞれの暗号化鍵は、個々の <xenc:EncryptedData> 内の <ds:KeyInfo> で指定することができる。

<xenc:ReferenceList> 下位要素が役に立つ典型的な状況は、作成者と受信者が共有された秘密鍵を利用するときである。この下位要素の利用例を次に示す。

```
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
xmlns:ds="..." xmlns:xenc="...">
    <S11:Header>
        <wsse:Security>
            <xenc:ReferenceList>
                <xenc:DataReference URI="#bodyID"/>
            </xenc:ReferenceList>
        </wsse:Security>
```

```
    </S11:Header>
    <S11:Body>
        <xenc:EncryptedData Id="bodyID">
          <ds:KeyInfo>
            <ds:KeyName>CN=Hiroshi Maruyama, C=JP</ds:KeyName>
          </ds:KeyInfo>
          <xenc:CipherData>
            <xenc:CipherValue>...</xenc:CipherValue>
          </xenc:CipherData>
        </xenc:EncryptedData>
    </S11:Body>
</S11:Envelope>
```

9.2 xenc:EncryptedKey

暗号化ステップが、SOAP エンベロープ内の要素または要素内容を対称鍵 (この鍵は受信者の鍵で暗号化されメッセージ中に埋め込まれる) による暗号化を含むとき、<xenc:EncryptedKey> がそのような暗号化された鍵を運ぶために利用されても**よい (MAY)**。この下位要素は、受信者がこの鍵で復号化される部分を知るために、目録、すなわち <xenc:ReferenceList> 要素をもつ**必要がある (SHOULD)**。この暗号化ステップで暗号化される要素または要素内容は、XML Encryption にしたがって、対応する <xenc:EncryptedData> によって置換されなければ**ならない (MUST)**。この暗号化ステップにより作られたすべての <xenc:EncryptedData> 要素は、この下位要素内の <xenc:ReferenceList> 要素に列挙される**必要がある (SHOULD)**。

この構成は、暗号化が無作為に生成された対称鍵で行われるとき (その鍵は続いて受信者の公開鍵によって暗号化される) に便利である。この要素の利用例を次に示す。

```
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
xmlns:ds="..." xmlns:xenc="...">
    <S11:Header>
        <wsse:Security>
              <xenc:EncryptedKey>
                ...
                <ds:KeyInfo>
                  <wsse:SecurityTokenReference>
                    <ds:X509IssuerSerial>
                      <ds:X509IssuerName>
                        DC=ACMECorp, DC=com
                       </ds:X509IssuerName>
<ds:X509SerialNumber>12345678</ds:X509SerialNumber>
                    </ds:X509IssuerSerial>
                  </wsse:SecurityTokenReference>
                </ds:KeyInfo>
                 ...
              </xenc:EncryptedKey>
  ...
        </wsse:Security>
    </S11:Header>
    <S11:Body>
        <xenc:EncryptedData Id="bodyID">
            <xenc:CipherData>
              <xenc:CipherValue>...</xenc:CipherValue>
            </xenc:CipherData>
        </xenc:EncryptedData>
    </S11:Body>
```

```
</S11:Envelope>
```

XML Encryption は <xenc:EncryptedKey> 要素が <xenc:EncryptedData> の中で指定されてもよい **(MAY)** と規定されているが、本規定では <xenc:EncryptedKey> が <wsse:Security> ヘッダに置かれることを強く**推奨する (RECOMMENDS)**。

9.3 処理規則

先に定義された下位要素の 1 つを利用して暗号化された部分は、XML Encryption 規定に準拠していなければ**ならない (MUST)**。暗号化された SOAP エンベロープは、なお妥当な SOAP エンベロープでなければ**ならない (MUST)**。メッセージ作成者は、<S11:Envelope> 要素、<S12:Envelope> 要素、<S11:Header> 要素、<S12:Header> 要素、または <S11:Body> 要素、<S12:Body> 要素を暗号化**してはならない (MUST NOT)** が、<S11:Header> 要素、<S12:Header> 要素、および <S11:Body> 要素または <S12:Body> 要素の子要素を暗号化しても**よい (MAY)**。同じ受信者を対象としているなら、暗号化の複数のステップが単一の <wsse:Security> ヘッダ部に追加されても**よい (MAY)**。

SOAP エンベロープ内の要素または要素内容 (例えば、<S11:Body> 要素または <S12:Body> 要素の内容) が暗号化されるとき、それは XML Encryption にしたがって <xenc:EncryptedData> に置換されなければ**ならず (MUST)**、この暗号化ステップで作られた <xenc:ReferenceList> 要素から参照される**必要がある (SHOULD)**。

9.3.1 暗号化

本仕様に準拠した暗号化された SOAP メッセージを作成するための一般的なステップ (規範ではない) を、以下に列挙する (<xenc:ReferenceList> の利用が**推奨される (RECOMMENDED)** ことに注意)。

- 新しい SOAP エンベロープを作成する。
- <wsse:Security> ヘッダを作成する。
- <xenc:EncryptedKey> が利用される場合、<wsse:Security> 要素の <xenc:EncryptedKey> 下位要素を作成する。この <xenc:EncryptedKey> 下位要素は、その鍵を利用して暗号化された各 <xenc:EncryptedData> 要素への <xenc:DataReference> を含んでいる <xenc:ReferenceList> 下位要素を含む**必要がある (SHOULD)**。
- 暗号化される項目 (すなわち対象とする SOAP エンベロープ内の XML 要素、および要素内容) の場所を見つける。
- 次のようにデータ項目を暗号化する。対象とする SOAP エンベロープ内のそれぞれの XML 要素または要素内容に対して、XML Encryption 規定 [XMLENC] の処理規則にしたがって暗号化する。それぞれの選択された元の要素または要素内容は削除され、結果として出てくる <xenc:EncryptedData> 要素により置換されなければ**ならない (MUST)**。
- <xenc:EncryptedData> 要素中の付加的な <ds:KeyInfo> 要素が他の <ds:KeyInfo> 要素を参照しても**よい (MAY)**。もし暗号化が添付されたセキュリティ・トークンに基づいているなら、その位置を見つけるために<wsse:SecurityTokenReference> 要素が <ds:KeyInfo> 要素に追加される**必要がある (SHOULD)**。

- 生成された <xenc:EncryptedData> 要素を参照する <xenc:DataReference> 要素を作成する。作成された <xenc:DataReference> を <xenc:ReferenceList> に追加する。
- すべての暗号化されていないデータを複製する。

9.3.2 復号

暗号化ヘッダ要素を含む SOAP エンベロープを受けとった際、各暗号化ヘッダ要素に対して次の一般的なステップが処理される必要がある (規範でない)。

受信者の所有となっている復号鍵を識別し、それから復号できるメッセージ要素を識別する。

(おそらくは <xenc:ReferenceList> を用いて)復号されるべき <xenc:EncryptedData> 項目の位置を見つける。

それらを次のように復号する。

- 目的の SOAP エンベロープの各要素に対して、XML Encryption 規定の処理規則と先に列挙した処理規則にしたがって復号する。
- 復号が何らかの利用で失敗したならば、アプリケーションは本規定の 12 章 エラー処理で定義された誤り符号を利用して作成者に失敗を報告しても**よい (MAY)**。

異なる Role (役割)で動作している SOAP ノードにより復号されることが意図されるような方法で、SOAP メッセージの重なる部分を暗号化することが可能である。この場合、これらの暗号化操作を識別する <xenc:ReferenceList> または <xenc:EncryptedKey> 要素は、必然的に異なる <wsse:Security> ヘッダに現れる。 SOAP は異なる Role がそのそれぞれのヘッダを処理する順序を指定する方法を提供しないため、この順序は本規定では規定されず、事前の同意によってのみ決定される。

9.4 復号変換

<wsse:Security> ヘッダの順序付けセマンティクスは、署名が暗号化されたデータを対象にしているか、または、暗号化されていないデータを対象にしているかを決定するのに十分である。しかしながら、署名が 1 つの <wsse:Security> ヘッダに含まれており、暗号化データが他の <wsse:Security> ヘッダにある場合、正しい処理順序は明らかでないかもしれない。

作成者が、仲介者によって引き続き暗号化されても**よい (MAY)** メッセージに署名することを望むなら、作成者は復号の順序を厳密に指定するために Decryption Transform for XML Signature を利用しても**よい (MAY)**。

10セキュリティ・タイムスタンプ

受信者にとってセキュリティ・セマンティクスの*鮮度*を判断できることがしばしば重要である。場合によっては、セキュリティ・セマンティクスがあまりに古くて、受信者がそれを無視すると判断するかもしれない。

本仕様では、時刻を同期する機構は提供しない。時刻は信頼されている、または、ここで説明されない追加の機構が反射攻撃を防ぐために採用されていることを想定している。

本仕様では XML Schema で定義された xsd:dateTime 型によって時刻参照を定義し説明する。すべての時刻参照がこの型を利用することが**推奨される (RECOMMENDED)**。更に、すべての参照は UTC 時刻であることが**推奨される (RECOMMENDED)**。実装はうるう秒を指定する時刻を生成**してはならない (MUST NOT)**。しかしながら、もし他の時刻型が利用されるなら、(以下で説明される) ValueType 属性が時刻形式のデータ型を示すために指定されなければ**ならない (MUST)**。

要求者と受信者はミリ秒よりも高い精度の時刻をサポートする他のアプリケーションに頼らないほうがよい **(SHOULD NOT)**。

<wsu:Timestamp> 要素はメッセージ中のセキュリティ・セマンティクスの生成時刻および有効期限を表現するための機構を提供する。

すべての時刻は、XML Schema 型 (dateTime) により指定されるように UTC 形式でなければ**ならない (MUST)**。時刻が XML Schema 仕様に定義されたような時刻精度をサポートすることに注意されるべきである。

<wsu:Timestamp> 要素は <wsse:Security> ヘッダの子として指定され、1 つのヘッダにつき (すなわち、SOAP actor/role につき) 多くて一度存在してもよい。

要素内の順序は下に示された通りである。 <wsu:Timestamp> 要素内の要素の順序は固定であり、仲介者によって保たれなければ**ならない (MUST)**。

<wsu:Timestamp> 要素に対するスキーマの概要は次の通りである。

```
<wsu:Timestamp wsu:Id="...">
    <wsu:Created ValueType="...">...</wsu:Created>
    <wsu:Expires ValueType="...">...</wsu:Expires>
    ...
</wsu:Timestamp>
```

次に上のスキーマでリストされた属性と要素を説明する。

/wsu:Timestamp

これはメッセージのタイムスタンプを示すための要素である。

/wsu:Timestamp/wsu:Created

これはセキュリティ・セマンティクスの生成時刻を表わす。この要素はオプションであるが、<wsu:Timestamp> 要素中で一度だけ指定されることができる。 SOAP 処理モデル内では、生成はインフォセットが伝送のためにシリアライズされた瞬間である。メッセージの生成時刻は、その伝送時刻から実質的に異ならないほうがよい **(SHOULD NOT)**。時刻の差は最小にされるべきである。

/wsu:Timestamp/wsu:Expires

これはセキュリティ・セマンティクスの有効期限を表わす。これはオプションであるが、<wsu:Timestamp> 要素中で一度だけ指定することができる。有効期限を過ぎるとそのセキュリティ・セマンティクスがもはや有効でないことを要求者が表明する。受信者 (このメッセージを処理するだれでも) はセキュリティ・セマンティクスが有効期限を過ぎたメッセージは廃棄 (無視) することが強く**推奨される (RECOMMENDED)**。受信者が要求者にセキュリティ・セマンティクスが有効期限を過ぎたことを告げたい場合の Fault コード (wsu:MessageExpired) が提供されている。サービスはセキュリティ・セマンティクスが有効期限を過ぎたことを示すフォールトを発行しても**よい (MAY)**。

/wsu:Timestamp/{any}

これは、追加の要素が要素に与えられることを許す拡張機構である。認識されない要素は失敗を発生する**必要がある (SHOULD)**。

/wsu:Timestamp/@wsu:Id

このオプションの属性は、この要素 (タイムスタンプ) を参照するために利用されることができる XML Schema ID を指定する。これは、例えば、XML Signature の中でタイムスタンプを参照するために利用される。

/wsu:Timestamp/@{any}

これは、追加の属性が要素に与えられることを許す拡張機構である。認識されない属性は失敗を発生する**必要がある (SHOULD)**。

有効期限は要求者の時計による。有効期限を評価するために、受信者は要求者の時計と受信者の時計が同期していないかもしれないことを認識する必要がある。したがって、受信者は有効期限が受信者の時計ではなく送信者の時計からみて過去であるかどうかを評価することが要請されるため、受信者は要求者の時計に対する信頼のレベルを評価をしなければ**ならない (MUST)**。受信者は本仕様で説明されない方法、例えば何らかの時計同期プロトコルによって要求者の現在の時計の時刻を推定してもよい。受信者は時計のズレの度合を見積もるために、生成時刻と仲介の SOAP role による遅延を利用してもよい。

次の例が <wsu:Timestamp> 要素とその内容の利用を示す。

```
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="...">
  <S11:Header>
    <wsse:Security>
      <wsu:Timestamp wsu:Id="timestamp">
         <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
         <wsu:Expires>2001-10-13T09:00:00Z</wsu:Expires>
      </wsu:Timestamp>
      ...
    </wsse:Security>
    ...
  </S11:Header>
  <S11:Body>
    ...
```

```
  </S11:Body>
</S11:Envelope>
```

11 拡張例

次のサンプル・メッセージはセキュリティ・トークン、署名、および暗号の利用を示す。この例では、タイムスタンプとメッセージ本体が暗号の前に署名されている。 <wsse:Security> ヘッダ内に署名/暗号の順序が指定されているため、decryption 変換は必要とされない。

```
(001) <?xml version="1.0" encoding="utf-8"?>
(002) <S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
xmlns:xenc="..." xmlns:ds="...">
(003)   <S11:Header>
(004)      <wsse:Security>
(005)         <wsu:Timestamp wsu:Id="T0">
(006)           <wsu:Created>
(007)                    2001-09-13T08:42:00Z</wsu:Created>
(008)         </wsu:Timestamp>
(009)
(010)         <wsse:BinarySecurityToken
                     ValueType="...#X509v3"
                     wsu:Id="X509Token"
                     EncodingType="...#Base64Binary">
(011)         MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh5i...
(012)         </wsse:BinarySecurityToken>
(013)         <xenc:EncryptedKey>
(014)             <xenc:EncryptionMethod Algorithm=
                        "http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
(015)             <ds:KeyInfo>
(016)                <wsse:KeyIdentifier
                         EncodingType="...#Base64Binary"
                      ValueType="...#X509v3">MIGfMa0GCSq...
(017)                </wsse:KeyIdentifier>
(018)             </ds:KeyInfo>
(019)             <xenc:CipherData>
(020)                <xenc:CipherValue>d2FpbmdvbGRfE0lm4byV0...
(021)                </xenc:CipherValue>
(022)             </xenc:CipherData>
(023)             <xenc:ReferenceList>
(024)                 <xenc:DataReference URI="#enc1"/>
(025)             </xenc:ReferenceList>
(026)         </xenc:EncryptedKey>
(027)         <ds:Signature>
(028)            <ds:SignedInfo>
(029)               <ds:CanonicalizationMethod
                    Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(030)               <ds:SignatureMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
(031)                  <ds:Reference URI="#T0">
(032)                     <ds:Transforms>
(033)                        <ds:Transform
                    Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(034)                     </ds:Transforms>
(035)                     <ds:DigestMethod
                     Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(036)                     <ds:DigestValue>LyLsF094hPi4wPU...
(037)                      </ds:DigestValue>
```

```
(038)               </ds:Reference>
(039)               <ds:Reference URI="#body">
(040)                  <ds:Transforms>
(041)                     <ds:Transform
                   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(042)                  </ds:Transforms>
(043)                  <ds:DigestMethod
                    Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(044)                  <ds:DigestValue>LyLsF094hPi4wPU...
(045)                   </ds:DigestValue>
(046)               </ds:Reference>
(047)            </ds:SignedInfo>
(048)            <ds:SignatureValue>
(049)                    Hp1ZkmFZ/2kQLXDJbchm5gK...
(050)            </ds:SignatureValue>
(051)            <ds:KeyInfo>
(052)                <wsse:SecurityTokenReference>
(053)                    <wsse:Reference URI="#X509Token"/>
(054)                </wsse:SecurityTokenReference>
(055)            </ds:KeyInfo>
(056)         </ds:Signature>
(057)      </wsse:Security>
(058)   </S11:Header>
(059)   <S11:Body wsu:Id="body">
(060)      <xenc:EncryptedData
                    Type="http://www.w3.org/2001/04/xmlenc#Element"
                    wsu:Id="enc1">
(061)         <xenc:EncryptionMethod
              Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-
cbc"/>
(062)         <xenc:CipherData>
(063)            <xenc:CipherValue>d2FpbmdvbGRfE0lm4byV0...
(064)            </xenc:CipherValue>
(065)         </xenc:CipherData>
(066)      </xenc:EncryptedData>
(067)   </S11:Body>
(068) </S11:Envelope>
```

この例の鍵となる部分をいくつか検査してみよう。

行 (003)-(058) は SOAP メッセージ・ヘッダを含む。

行 (004)-(057) は <wsse:Security> ヘッダ部を表わす。これはメッセージに対するセキュリティ関連の情報を含む。

行 (005)-(008) はタイムスタンプ情報を指定する。この場合、セキュリティ・セマンティクスの生成時刻を示す。

行 (010)-(012) はメッセージに関連付けられたセキュリティ・トークンを指定する。この場合、Base64 に符号化された X.509 証明書を指定する。行 (011) は証明書の実際の Base64 符号化したものを指定する。

行 (013)-(026) はメッセージの本体を暗号化するために利用された鍵を指定する。これは対称鍵であるので、暗号化された形式で渡されている。行 (014) は鍵を暗号化するために利用されたアルゴリズムを定義する。行 (015)-(018) は対称鍵を暗号化するために利用された鍵の識別子を指定する。行 (019)-(022) は対称鍵の実際の暗号化された形式を指定する。行 (023)-(025) はこの対

称鍵を利用したメッセージ中の暗号化部分を識別する。この場合、本体 (ID="enc1") の暗号化に利用されただけである。

行 (027)-(056) はデジタル署名を指定する。この例では、署名は X.509 証明書を基にしている。行 (028)-(047) は何が署名されているかを示している。特に、行 (039) はメッセージ本体を参照している。

行 (048)-(050) は実際の署名値 - 行 (043) で指定される - を示している。

行 (052)-(054) は署名に利用された鍵を示している。この場合、メッセージに含まれた X.509 証明書である。行 (053) は行 (010)-(012) へリンクする URI を提供する。メッセージの本体は行 (057)-(067) で表現されている。

行 (060)-(066) は暗号化されたメタデータと XML Encryption を利用した本体の形式を表現する。

行 (059) は "要素値" が置き換えられたことを示し、この暗号化を識別する。行 (061) は暗号アルゴリズム - この場合 Triple-DES - を指定する。行 (063)-(064) は実際の暗号テキスト (すなわち、暗号化の結果) を含む。鍵がこの暗号化を参照しているため - 行 (024) 、鍵への参照を含まないことに注意のこと。

12 エラー処理

セキュリティ情報を処理する間にエラーが起りうるような多くの状況が存在する。例えば、

- 無効な、または、サポートされてないセキュリティ・トークン、署名または暗号
- 無効な、または認証されない、または認証不可能なセキュリティ・トークン
- 無効な署名
- 復号の失敗
- 参照されたセキュリティ・トークンが利用可能でない
- サポートされない名前空間

Security ヘッダの内容のためにサービスが通常の操作を遂行できない場合、SOAP の Fault Mechanism を利用してそのことが報告されても**よい (MAY)**。本規定では、これはサービス拒否や暗号の攻撃の部分として利用されることができるので、失敗が返却されることを命令しない。署名と暗号の失敗をある型の攻撃を柔らげるために組み合わせる。

失敗が製作者に返される場合、失敗は SOAP の Fault 機構を利用して報告されなければ**ならない (MUST)**。次の表に事前に定義されたセキュリティ失敗符号を概観する。 "unsupported" クラスのエラーは次の通りである。以下で提供される理由の文は**推奨される (RECOMMENDED)** が、実装により、より記述的なまたは好ましいならば、代替の文が提供されても**よい (MAY)**。以下の表は SOAP 1.1 の用語で定義されている。 SOAP 1.2 に対しては、Fault/Code/Value は (SOAP 1.2 で定義されるように) env:Sender と Fault/Code/Subcode/Value は以下の faultcode であり、Fault/Reason/Text は以下の faultstring である。

発生したエラー (faultstring)	Faultcode
サポートされないトークンが与えられた	wsse:UnsupportedSecurityToken
サポートされない署名または暗号アルゴリズムが利用された	wsse:UnsupportedAlgorithm

"failure" クラスのエラーは次の通りである。

発生したエラー (faultstring)	faultcode
<wsse:Security> ヘッダの処理でエラーが発見された	wsse:InvalidSecurity
無効なセキュリティ・トークンが提供された	wsse:InvalidSecurityToken
セキュリティ・トークンが認証されなかった、または、認可されなかった	wsse:FailedAuthentication
署名または復号が無効であった	wsse:FailedCheck
参照されたセキュリティ・トークンが取得される	wsse:SecurityTokenUnavailable

ことができなかった	

13セキュリティの考慮

本文書の目標と要求で述べたように、本規定は様々なセキュリティ機構を実装できるような、拡張可能な枠組みと柔軟な構文を提供することが意図されている。この枠組みと構文はそれ自信ではセキュリティの保証を提供しない。この枠組みと構文を実装して利用するときに、結果が幅広い攻撃のどれに対しても脆弱でないということを保証するために、あらゆる努力がされなければならない。

13.1 一般的な考慮

そのような拡張可能な機構の集合に対して、セキュリティの考慮の包括的なリストを提供することは可能ではない。完全なセキュリティ解析は、本規定を基にした特定のソリューションに実施されなければ**ならない (MUST)**。以下に、この型のプロトコルでしばしば持ち上がるセキュリティの懸念のいくつかを説明するが、これが懸念の完全なリストではないことを強調する。

- 新鮮さの保証 (例えば、再送の危険、遅延したメッセージ、および、安全な時刻同期を仮定したタイムスタンプに頼ることへの危険)
- デジタル署名と暗号の適切な利用 (メッセージの重要な部分に署名/暗号化すること、署名と暗号化の間の相互操作)、すなわち、平文にあるとき、暗号化されたメッセージ (の内容) への署名は情報を漏らす)
- セキュリティ・トークンの保護 (完全性)
- 証明書検証 (廃止の問題も含めて)
- 外部の保護なしにパスワードを利用する危険 (すなわち、パスワードに対する辞書攻撃、再送、パスワードで導出された鍵の不安全、...)
- 乱数の利用 (または強い擬似乱数)
- 本標準と他のシステム構成要素を実装するセキュリティ機構間の相互操作
- 中間者攻撃
- PKI 攻撃 (すなわち、アイデンティティ混乱)

セキュリティ・プロトコルで考慮する必要があるかもしれない他のセキュリティ懸念がある。上のリストは、包括的なセキュリティ解析の代わりに「チェック・リスト」として利用されるべきではない。次の節では、このリスト中の考慮のいくつかの詳細を少し提供する。

13.2 追加の考慮

13.2.1 再送

デジタル署名はそれだけではメッセージ認証を提供しない。誰かが署名付きメッセージを記録して、再送することがありうる (再送攻撃)。開かれたネットワークを経由してメッセージが交換される場合、メッセージ受信者がメッセージ再送を検知できるように、メッセージの中にデジタル署名された要素が含まれることが強く**推奨される (RECOMMENDED)**。これら (署名される要素) はメッセージまたは他の SOAP 拡張で定義されたヘッダの部分とすることができる。典型的なものとして次の 4 つである: タイムスタンプ、連続番号、有効期限、および、メッセ

ージ相関関係。タイムスタンプが与えられた時間の間キャッシュされることが**推奨される (RECOMMENDED)**。Signed timestamps MAY be used to keep track of messages (possibly by caching the most recent timestamp from a specific service) and detect replays of previous messages. ガイドラインとして 5 分という値が再送を検知するための最小として利用されることができる。その与えられた時間の間よりも古いタイムスタンプは対話式のシナリオでは拒否されるべきであることを。

13.2.2 セキュリティ機構の組合せ

本規定は SOAP ヘッダ内での XML Signature と XML Encryption の利用について定義する。SOAP メッセージを安全にするための構成要素の 1 つとして、それ (本規定) は他のセキュリティ技術とともに利用されることが意図されている。デジタル署名は他のセキュリティ機構と実体へ起こりうる脅威という文脈で理解される必要がある。

実装者はまた、一般にデジタル署名の利用から、特に XML Signature の利用から結果として起こるセキュリティの含意をすべて知っているべきである。デジタル署名を基にしてアプリケーションに信頼を構築するとき、証明書評価のように組み込まれるべき他の技術があるが、これらは本文書の範囲外である。

XML Encryption で説明されているように、共通のデータ項目に渡る署名と暗号の組合せがある暗号的脆弱性を導入するかもしれないことに注意する。例えば、デジタル署名を明らかにしたままにして、デジタル的に署名されたデータを暗号化することは、平文推測攻撃を許すかもしれない。

13.2.3 チャレンジ

デジタル署名が送信実体に付随する申告を検証するために利用されるとき、製作者は確認鍵の知識を誇示しなければならない。これを達成する 1 つの方法は、挑戦-応答型のプロトコルを利用することである。そのようなプロトコルは本文書の範囲外である。この目的のために、開発者はタイムスタンプ、有効期限、および連番をメセージに付加することができる。

13.2.4 セキュリティ・トークンと鍵の保護

実装者はトークン置き換え攻撃の可能性も知っておくべきである。デジタル署名がメッセージ (それは鍵を指定する) 中に提供されるトークンへの参照によって検証されるどの状況でも、無法な製作者が後に同じ鍵を含むが、異なる情報が意図された、異なるトークンを申告することが可能かもしれない。

これの例は、同じ鍵の組に関連して発行されたが、異なる属性、制約、または信頼限度をもつ複数の X.509 証明書をもっていた利用者であろう。発行権威者によるトークンへの署名はこの攻撃を防がないことに注意すること。また、利用者が秘密の保持を証明することができるなら、権威者は異なる権威者が同じ鍵にたいしてトークンを発行することを効果的に防ぐこともできない。

この攻撃への最も直接的な対抗策は、トークン (または、その一意の識別するデータ) が製作者の署名の下に含まれることを主張することである。アプリケーションの本質は、それが信頼された権威者により発行されたと仮定して、トークンの内容が不適切であるということなら、この攻撃は無視されるかもしれない。しかしながら、アプリケーションのセマンティクスが時間に渡って変化するかもしれないので、最良の慣習はこの攻撃を防御することである。

1564 要求者は、それらが転送中に改変されなかったことを保証するために、署名 (または、他の保
1565 護機構) を含まないセキュリティ・トークンに署名するためにデジタル署名を利用するべきで
1566 ある。すべての関連した不変のメッセージ内容が製作者によって署名されることを強く**推奨す**
1567 **る (RECOMMENDED)**。受信者は、製作者の署名により覆われている文書の部分を、メッセー
1568 ジ中のセキュリティ・トークンの対象となると考慮するだけである**必要がある (SHOULD)**。
1569 <wsse:Security> ヘッダ要素に現れるセキュリティ・トークンは、メッセージ受信者がセキュ
1570 リティ・トークンが発行されてから偽造または改変されてないことを確信することができるよ
1571 う、発行権威者によって署名される**必要がある (SHOULD)**。メッセージ製作者は、それが確認
1572 していて、発行権威者によって署名されてないどの <SecurityToken> 要素にも署名することを
1573 強く**推奨される (RECOMMENDED)**。

1574 要求者が、要求の中に、返答を暗号化するために利用されるべき公開鍵を提供するとき、中間
1575 にいる攻撃者が異なる公開鍵に置き換え、攻撃者が返答を読むことを許してしまうことが可能
1576 である。この攻撃を防御する最良の方法は、暗号鍵を何らかの方法で要求に結び付けることで
1577 ある。これをする 1 つの単純な方法は、要求に署名するのと、応答を暗号化するのと同じ鍵の
1578 組を利用することである。しかしながら、もしポリシーが署名と暗号に異なる鍵の組を利用す
1579 ることを要求するならば、要求で提供される公開鍵は要求の署名の下に含まれるべきである。

1580 13.2.5 タイムスタンプと Id の保護

1581 wsu:Id 属性と <wsu:Timestamp> 要素を信頼するために、それらは本規定で概観した機構を利
1582 用して署名される**必要がある (SHOULD)**。このことは、ID とタイムスタンプ情報の読者が、
1583 ID とタイムスタンプが何らかの方法で偽造または改変されてないことを確かにすることを許
1584 す。 ID とタイムスタンプ要素が署名されることを強く**推奨される (RECOMMENDED)**。

1585

1586 本節は参考である。

14 相互運用の注記

本規定および類似の規定での相互運用の経験に基づいて、次のリストが相互運用性の問題が発見されたいくつかの共通の領域を強調する。これらの問題を避けるために実装時には注意が払われるべきである。これらのいくつかは "明白な" に思えるかもしれないが、テストを通して問題があったことに注目されるべきである。

鍵識別子: アルゴリズムとそれがどのようにセキュリティ・トークンに適用されているかを理解していることを確実にすること。

EncryptedKey: XML Encryption からの <xenc:EncryptedKey> 要素は、値の既定義リストの 1 つの値をもつ Type 属性を要求する。正しい値が利用されていることを保証すること。

暗号化のパッディング: XML Encryption のランダム・ブロック暗号パッディングがある復号実装で問題を引き起こした。仕様に正確にしたがうように注意深いこと。

ID 群: 規定は 3 つの特定の ID 要素を認識する: グローバルな wsu:Id 属性と XML Signature と XML Encryption 要素のローカルな ID 属性 (後ろ 2 つはグローバルな属性を許さないため)。もし他の要素がグローバルな属性を許さないなら、ID 参照を利用して直接署名することができない。グローバル属性 wsu:Id が名前空間規定を支えなければ**ならない (MUST)** ことに注意すること。

時刻の形式: 本規定では XML Schema xsd:dateTime 要素の制限された版を利用する。指定された制限に準拠を保証するよう気をつけること。

バイト・オーダー・マーカ (BOM): いくつかの時相では BOM マーカの処理で問題があった。これの利用はオプションであることが示唆される。

SOAP、WSDL、HTTP: さまざまな相互運用の問題が、採用されている不正確な SOAP、WSDL、および HTTP セマンティクスに見られた。これらの規定と利用可能な相互運用ガイドラインに注意深く固執することに注意すること。

本節は参考である。

15 プライバシの考慮

本規定の文脈では、ここで定義されたセキュリティ要素による潜在的なプライバシ侵害に関心があるだけである。ペイロード・メッセージの内容のプライバシは範囲外である。製作者または送信するアプリケーションは、セキュリティ・トークンに集められた申告が典型的には個人的な情報であり、そのため製作者のプライバシ・ポリシーにしたがって送られるのみであるべきである。将来の標準がプライバシの義務または制限がこのデータに加えられることを許すかもしれない。そのような標準が利用されるのでなければ、製作者は受信者がデータに関連したすべての宣言に固執することに結び付けられることを out-of-band で保証しなければならず、受信者は同様に out-of-band の方法によってデータの意図された処理のための必要な了解をもつことを保証しなければならない。

申告データが仲介者に可視であるなら、ポリシーはまたこれらの仲介者への開放を許さなければならない。もっとも個人的な情報は、任意の関係者へ開放されることはできないので、これは動作者が識別できる方法で参照されることを典型的に要求するだろう。そのような身元を確認できる参照はまた典型的に仲介者の適切な暗号鍵を獲得するために必要とされる。仲介者が申告を追加するなら、本来の製作者のようなプライバシ・ポリシーによって指針とされるべきである。

仲介者はまた、SOAP メッセージ交換からトラフィックの情報を得るかもしれない。例えば、誰が誰といつ通信しているか。仲介者を使う製作者は、そのプライバシ・ポリシーに適合する選ばれた仲介者へ、このトラフィック情報を開放することを検証すべきである。

本節は参考である。

16参考文献

[GLOSS] Informational RFC 2828, "Internet Security Glossary," May 2000.

[KERBEROS] J. Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5)," RFC 1510, September 1993, http://www.ietf.org/rfc/rfc1510.txt .

[KEYWORDS] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997

[SHA-1] FIPS PUB 180-1. Secure Hash Standard. U.S. Department of Commerce / National Institute of Standards and Technology. http://csrc.nist.gov/publications/fips/fips180-1/fip180-1.txt

[SOAP11] W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.

[SOAP12] W3C Recomendation**, "**http://www.w3.org/TR/2003/REC-soap12-part1-20030624/", 24 June 2003

[SOAPSEC] W3C Note, "SOAP Security Extensions: Digital Signature," 06 February 2001.

[URI] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.

[XPATH] W3C Recommendation, "XML Path Language", 16 November 1999

次のものは、経緯と関連する資料のために含まれる参考的なものである。

[WS-SECURITY] "Web Services Security Language", IBM, Microsoft, VeriSign, April 2002.
"WS-Security Addendum", IBM, Microsoft, VeriSign, August 2002.
"WS-Security XML Tokens", IBM, Microsoft, VeriSign, August 2002.

[XMLC14N] W3C Recommendation, "Canonical XML Version 1.0," 15 March 2001

[EXCC14N] W3C Recommendation, "Exclusive XML Canonicalization Version 1.0," 8 July 2002.

[XMLENC] W3C Working Draft, "XML Encryption Syntax and Processing," 04 March 2002

[XML-ns] W3C Recommendation, "Namespaces in XML," 14 January 1999.

[XMLSCHEMA] W3C Recommendation, "XML Schema Part 1: Structures,"2 May 2001.
W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001.

[XMLSIG] W3C Recommendation, "XML Signature Syntax and Processing," 12 February 2002.

[X509] S. Santesson, et al,"Internet X.509 Public Key Infrastructure Qualified Certificates Profile,"
http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T-REC-X.509-200003-I

[WSS-SAML] OASIS Working Draft 06, "Web Services Security SAML Token Profile", 21 February 2003

[WSS-XrML] OASIS Working Draft 03, "Web Services Security XrML Token Profile", 30 January 2003

[WSS-X509] OASIS, “Web Services Security X.509 Certificate Token Profile”, 19 January 2004, http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0

[WSSKERBEROS] OASIS Working Draft 03, "Web Services Security Kerberos Profile", 30 January 2003

1676 **[WSSUSERNAME]** OASIS,"Web Services Security UsernameToken Profile" 19 January
1677 2004, http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
1678 username-token-profile-1.0
1679 **[WSS-XCBF]** OASIS Working Draft 1.1, "Web Services Security XCBF Token Profile",
1680 30 March 2003
1681 **[XPOINTER]** "XML Pointer Language (XPointer) Version 1.0, Candidate
1682 Recommendation", DeRose, Maler, Daniel, 11 September 2001.

Appendix A. ユーティリティ要素と属性

本規定では、他の規定により再利用されることのできるいくつかの要素、属性、および属性グループを定義している。この付録では、これらのユーティリティ構成要素の概要を提供する。詳細の説明が本規定で提供され、この付録では規定で文書化されてない他の側面を呼び出すと同様にこれらの節を参照することに注意されるべきである。

A.1. 識別子属性

SOAP メッセージ中の要素が参照されることを必要とする多くの状況がある。例えば、SOAP メッセージに署名するとき、選択された要素が署名に含まれる。 XML Schema Part 2 では要素を識別し参照するために利用されるかもしれないいくつかのビルトインのデータ型を提供しているが、それらの利用には SOAP メッセージの消費者が身元または参照機構が定義されているスキーマをもっているまたは獲得できることが要求される。いくつかの環境、例えば、仲介者では、これは問題を引き起こし、望ましくないことに成り得る。

したがって、SOAP 基盤を基にして、要素を識別し参照するために機構が必要とされる。それは要素が利用される文脈の完全なスキーマ知識に頼らない。この機能は、動的なスキーマ発見と処理なしに要素が識別され参照されることができるよう、SOAP 処理装置へと統合されることができる。

本規定では、任意の属性を許すまたは特にこの属性を許す要素に適用されることができる、要素を識別するための名前空間限定のグローバル属性を規定する。これは必要に応じて再利用さっることができる一般目的の機構である。

詳細の説明は 4.0 章 ID 参照に見つけられることができる。

本節は参考である。

A.2. タイムスタンプ要素

本規定では、生成および期限切れのようなタイムスタンプ情報を表現するために利用されてもよい XML 要素を定義する。メッセージのセキュリティの文脈で定義されているが、これらの種の時刻記述が作られることが必要とされるどこでも、これらの要素は再利用されることができる。

本規定での要素は XML Schema で定義された dateTime 型によって時刻参照を利用して定義され説明される。相互運用性のあめに、すべての時刻参照はこの型を利用することが**推奨される (RECOMMENDED)**。It is further RECOMMENDED that all references be in UTC time for increased interoperability. しかしながら、他の時刻型が利用されるなら、時刻形式のデータ型を示すために ValueType 属性が指定されなければ**ならない (MUST)**。

次の表がこれらの要素の概要を提供する。

Element	説明
<wsu:Created>	この要素は、囲んでいる文脈と関連した生成時刻を示すために利

	用される。
<wsu:Expires>	この要素は、囲んでいる文脈と関連した期限切れ時刻を示すために利用される。

1715

1716 詳細の説明は 10 章に見つけることができる。

1717

1718 本節は参考である。

1719 A.3. 一般のスキーマ型

1720 本規定のユーティリティ側のスキーマはまた、いくつかの一般的な目的のスキーマ要素も定義す
1721 る。これらの要素は本規定とともの利用されるためにこのスキーマで定義されているが、それら
1722 は他の規定でも同様に利用されてもよいような一般目的の定義である。

1723 特に、次のスキーマ要素が定義されており、再利用されることができる。

スキーマ要素	説明
wsu:commonAtts 属性グループ	この属性グループは要素に対して推奨される共通の属性を定義する。これには wsu:Id 属性や他の名前空間限定された属性のための拡張性が含まれる。
wsu:AttributedDateTime 型	この型は共通属性を含めるために XML Schema dateTime 型を拡張する。
wsu:AttributedURI 型	この型は共通属性を含めるために XML Schema anyURI 型を拡張する。

1724

1725 本節は参考である。

Appendix B. SecurityTokenReference モデル

この付録は、<wsse:SecurityTokenReference> 要素のための利用方法と処理モデルの非規範的な概要を提供する。

<wsse:SecurityTokenReference> 要素を導入することに対していくつかの動機がある。

XLM Signature 参照機構は、一般のトークン参照よりも "鍵" 参照に焦点を置いている。

XML Signature 参照機構は、適用されることのできる拡張性を制限するかなり閉じたスキーマを利用する。

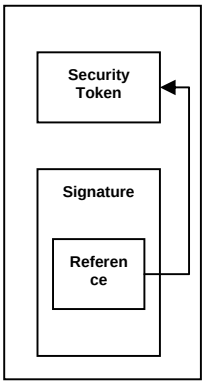
必要とされるが、XML Signature でカバーされない一般の参照機構の付加的な型がある。

参照が XML Signature の外側で起こるかもしれず、XML Signature のスキーマが適切でないまたは望まれないようなシナリオがある。

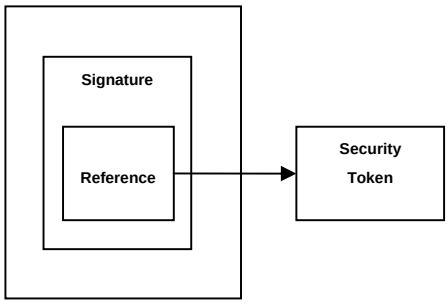
XML Signature 参照は、すべての参照へ適用しなくてもよい側面 (例えば、変換) を含むかもしれない。

次の利用事例が上の動機を駆り立てる。

ローカル参照 – <wsse:Security> ヘッダにメッセージに含められたセキュリティ・トークンが XML Signature に関連付けられる。下図にこれを示す。

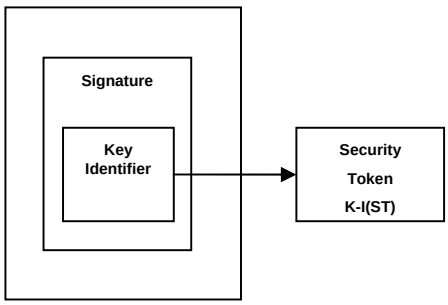


リモート参照 – メッセージに含まれないが特定の URI で利用可能かもしれないセキュリティ・トークンが XML Signature に関連付けられる。下図にこれを示す。



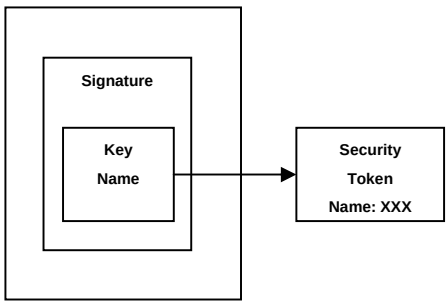
1745
1746

1747 **鍵識別子** – XML Signature に関連付けられ、セキュリティ・トークンのよく知られた関数の結
1748 果である知られた値を利用して識別されるセキュリティ・トークン (トークン形式またはプロ
1749 ファイルにより定義される) 下図がトークンが外部に位置している場合を示す。



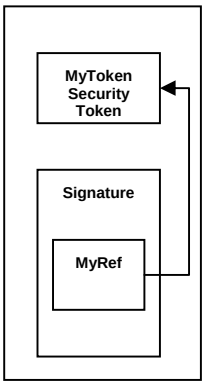
1750
1751

1752 **鍵名** – セキュリティ・トークンが XML Signature と関連付けられ、セキュリティ・トークン
1753 内部の "名前" 表面を表現する知られた値を利用して識別される (トークン形式またはプロファ
1754 イルにより定義される)。下図がトークンが外部に位置している場合を示す。

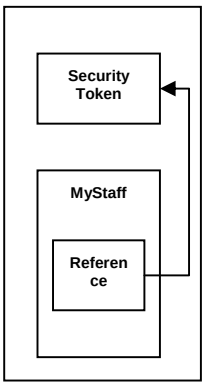


1755
1756

1757 **形式固有の参照** – セキュリティ・トークンが XML Signature に関連付けられ、(上で説明され
1758 た一般の機構ではなくて) トークンに固有の機構を利用して識別される。下図にこれを示す。



非署名の参照 – メッセージが XML 署名を表現しない XML を含むかもしれず、セキュリティ・トークンを参照するかもしれない (それはメッセージに含まれてもいなくてもよい)。下図にこれを示す。



すべての準拠する実装は <wsse:SecurityTokenReference> 要素を処理できなければ**ならない (MUST)**。しかしながら、参照の異なる型のすべてをサポートすることは要求されない。

参照は、望まれるトークンの型に対する "ヒント" を提供する ValueType 属性を含めても**よい (MAY)**。

複数の下位要素が指定されるなら、一緒にそれらはトークンのための参照を記述する。相互運用を試みるときに実装者が直面するいくつかの挑戦がある。

ID 参照 – XML 基底型の ID を利用した下にある XML 参照機構は、単純で直接的な XML 要素参照を提供する。しかしながら、これは XML 型であるので、どの属性に結び付けられることもできる。したがって、ID と参照を処理するためには、受信者がスキーマを理解することが要求される。特定の名前空間 URI に対する "スキーマの位置" を知る方法がないため、これは高価なタスクであり、一般の場合には不可能である。

あいまい性 – 参照の第一の目的は望まれるトークンを一意に識別することである。 ID 参照は、定義では、XML 毎に一意である。しかしながら、一意であるためには "主体の名前" のような他の機構は必要とされず、したがってそのような参照は一意でないかもしれない。

XML Signature 規定では、署名中で利用される "鍵" に関する情報を提供するために利用される <ds:KeyInfo> 要素を定義している。署名内のトークン参照に対しては、<wsse:SecurityTokenReference> が <ds:KeyInfo> 内に置かれることが**推奨される (RECOMMENDED)**。 XML Signature 規定はまた、識別子により、または、特定の鍵を渡すことにより鍵を参照するための機構を定義している。規則として、WSS: SOAP Message Security またはそのプロファイルで定義された特定の機構が XML Signature 中の機構よりも優先される。

次に、WSS: SOAP Message Security で定義される特定の参照機構の付加的な詳細を提供する。

直接参照 – <wsse:Reference> 要素がセキュリティ・トークンへの URI 参照を提供するために利用される。断片のみが指定さるなら、それは <wsu:Id> がその断片に適合する、文書中のセキュリティ・トークンを参照する。断片でない URI に対しては、参照は URI を利用して識別される [もしかすると外部にあるかもしれない] セキュリティ・トークンへのものである。 URI の処理の周りに暗に含まれたセマンティクスはない。

鍵識別子 – <wsse:KeyIdentifier> 要素はトークンのための既知の値 (識別子) を指定することによってセキュリティ・トークンを参照するために利用される。それはセキュリティ・トークンへ特別な関数を適用することによって決定される (例えば、鍵領域のハッシュ)。この手段は、特定のセキュリティ・トークンに対して典型的に一意であるが、プロファイルまたはトークン固有の関数が指定される必要がある。 ValueType 属性が鍵識別子の型を定義し、したがって、参照されるトークンの型を識別する。 EncodingType 属性は一意の値 (識別子) がどのように符号化されているかを指定する。例えば、ハッシュ値が base 64 符号化 (デフォルト) を利用して符号化されるかもしれない。

鍵名 – <ds:KeyName> 要素は、セキュリティ・トークン内の身元表明と適合するために利用される特定の値を指定することによってセキュリティ・トークンを参照するために利用される。これは部分集合適合であり、特定の名前に適合する複数のセキュリティ・トークンという結果になるかもしれない。 XML Signature はフォーマッティングのセマンティクスを含意しない一方、WSS: SOAP Message Security では X.509 名を指定することが推奨される **(RECOMMENDS)**。

適切なところでは、プロファイルが、参照機構が特定のトークン・プロファイルへとマップするかどうか、どのようにかを定義することが期待される。特に、プロファイルは次の質問に答えるべきである。

- 参照のどの型が利用されることができるか?
- "鍵名" 参照がどのようにマップするか (あるなら) ?
- "鍵識別子" 参照がどのようにマップするか (あるなら) ?
- 付加的なプロファイルまたは形式固有の参照があるか?

本節は参考である。

Appendix C. 改版履歴

Rev	Date	What
01	20-Sep-02	Initial draft based on input documents and editorial review
02	24-Oct-02	Update with initial comments (technical and grammatical)
03	03-Nov-02	Feedback updates
04	17-Nov-02	Feedback updates
05	02-Dec-02	Feedback updates
06	08-Dec-02	Feedback updates
07	11-Dec-02	Updates from F2F
08	12-Dec-02	Updates from F2F
14	03-Jun-03	Completed these pending issues - 62, 69, 70, 72, 74, 84, 90, 94, 95, 96, 97, 98, 99, 101, 102, 103, 106, 107, 108, 110, 111
15	18-Jul-03	Completed these pending issues – 78, 82, 104, 105, 109, 111, 113
16	26-Aug-03	Completed these pending issues - 99, 128, 130, 132, 134
18	15-Dec-03	Editorial Updates based on Issue List #30
19	29-Dec-03	Editorial Updates based on Issue List #31
20	14-Jan-04	Completed issue 241 and feedback updates
21	19-Jan-04	Editorial corrections for name space and document name
22	17-Feb-04	Editorial changes per Karl Best

Appendix D. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

OASIS は、本文書で記述された技術の実装や利用に関して主張される可能性がある知的財産や他の権利の正当性や範囲についてや、そのような権利のライセンスが利用可能または不可能かもしれないことについて、何の立場もとらないし、そのような権利を確認するために努力してきたとも主張しない。OASIS 仕様の権利に関する OASIS の手続き情報は OASIS ウェブサイトで見ることができる。公表のために利用可能となっている権利の主張の写しと利用可能となるであろうライセンスの保証、または、本仕様の実装者または利用者がそのような財産権を利用するための一般的なライセンスまたは許可を取得しようとした試みの結果は、OASIS Executive Director から取得することができる。
OASIS は、関心のあるものは誰でも、本仕様を実装するために必要とされる技術を対象とする著作権、特許または特許申請、または、他の財産権についての注意をうながしてもらうようお願いする。このような情報については、OASIS Executive Director に連絡してほしい。