MOF2008 合同デモシステム向けセキュリティ報告書

2008 年 9 月
XML コンソーシアム
セキュリティ部会

1. 概要

XML コンソーシアム セキュリティ部会では、2007 年の製造情報連携フォーラム SCF2007 デモシステム向けセキュリティ検討[1]に引き続き、MOF2008 (Manufactureing Open Forum 2008)の実証デモシステムにおけるセキュリティについて、検討を行った。

検討メンバー:

- 大沼啓希 (日本アイ・ビー・エム株式会社)
- 中山弘二郎 (株式会社日立製作所)
- 舟木康浩 (日本セーフネット株式会社)
- 松永豊 (東京エレクトロン デバイス株式会社)
- 宮地直人 (有限会社ラング・エッジ)

MOF2008 では、工場を想定して多種システムの連携をデモンストレーションする行動実証デモが行われる。

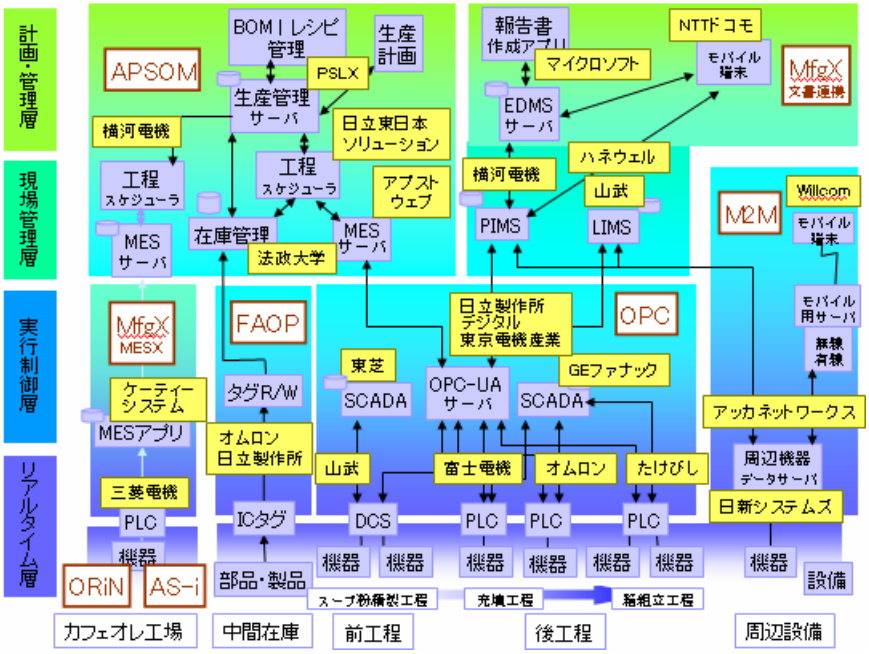


図 1 MOF2008 実証デモ構成図

1.1. 対象システム

2007 年の検討では、システム間連携を行うデモシステム全体に対して、必要となるセキュリティ要件を網羅的に検討したが、今回は実装面も含めた、より具体的な情報提供を目的に検討を行った。

今回のセキュリティ検討では、実証システム中のAPSOMが担当するプランナから生産管理サーバに送信される生産オーダ情報を対象とした。

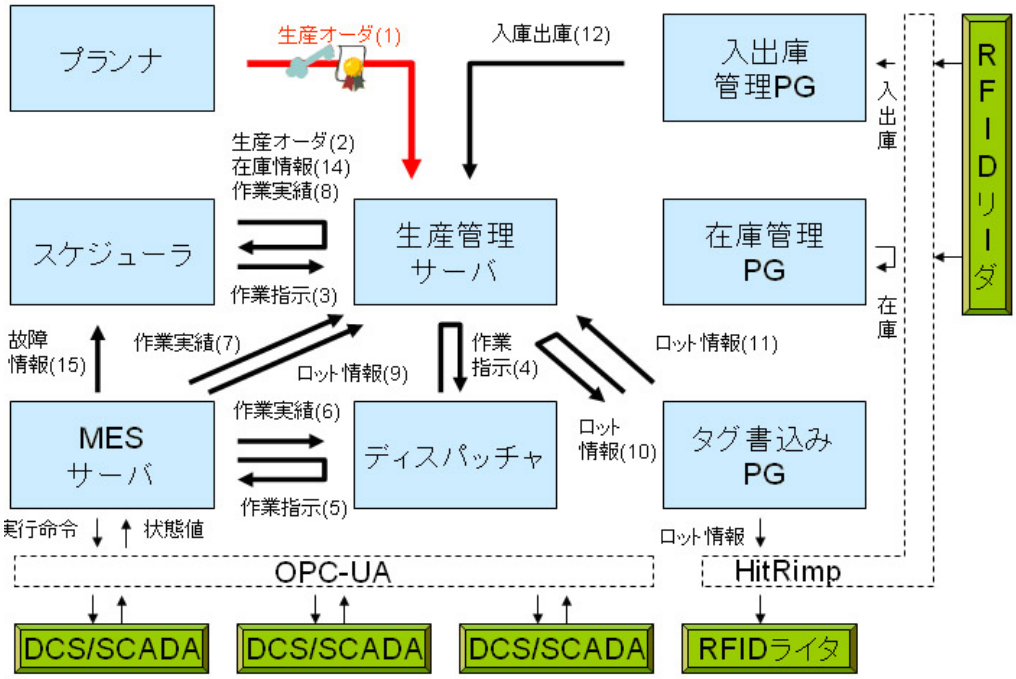


図 2 APSOM 部分の構成図

1.2. 検討の内容

生産オーダ情報に要求されるセキュリティ要件は、次のようなものが考えられる。

- 機密性 – 生産計画などを類推できる可能性があるため、外部に漏洩させたくない
- 完全性 – 生産指示の記録は正確性を要求されるため、改竄を防止したい

したがって、本検討では、機密性を確保するための暗号化と、完全性を確保するための電子署名について、それぞれ解説した上で、具体的な手段を検討し、手段ごとの特徴や課題などを報告する。

なお、手段として紹介するツールは、利用可能なツールを網羅するものではなく、検討メンバーがツールと利用のための情報を容易に入手できたものを例として取り上げている。

2. XML メッセージの保護に必要なセキュリティ技術

2.1. 概要

XMLデータに関するセキュリティ要件は多岐に渡る。そのため、近年、様々なXMLのセキュリティに関する技術が提案されている。図 3に標準化団体であるW3C (http://www.w3.org/)およびOASIS (http://www.oasis-open.org/)によって策定された主なXMLセキュリティ技術を示す（ここに示したものは一部であり、他にも数多くの技術が存在する）。以下本章では、これらの技術のうち、XMLデータの機密性および完全性の確保に関する技術であるXML暗号、XML署名、WS-Security、XAdESの概要を述べる。

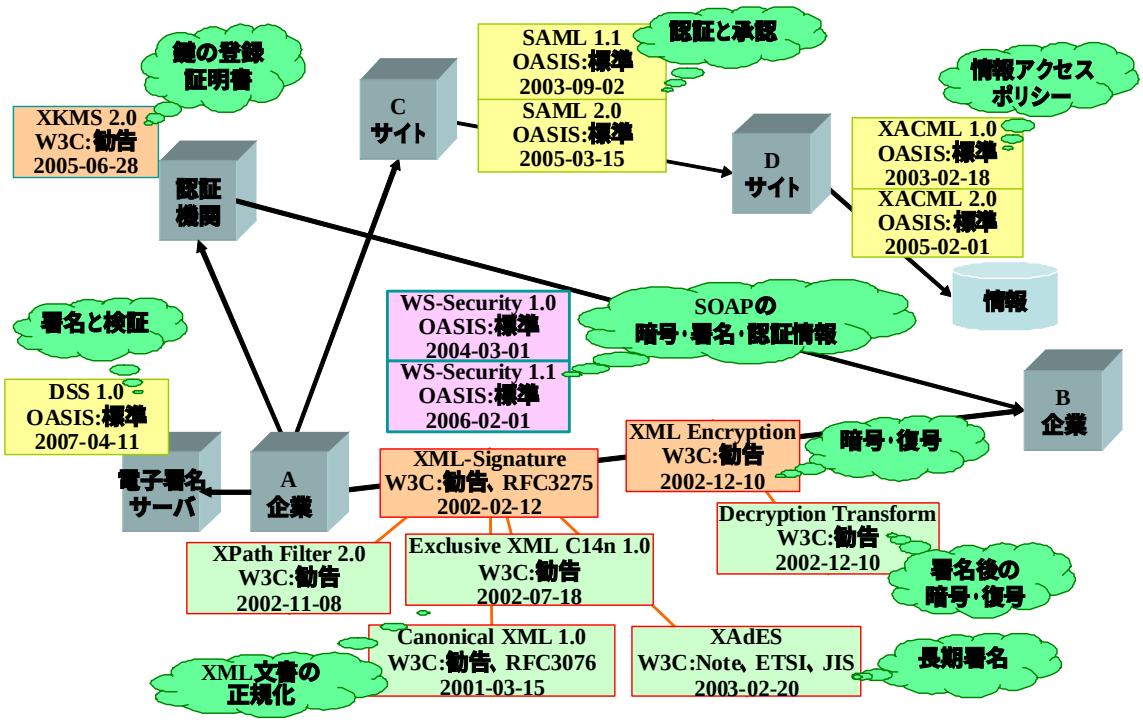


図 3 XML セキュリティ関連規格一覧

2.2. 暗号化（XML 暗号）

XML 暗号（XML Encryption Syntax and Processing）は、W3C の XML Encryption WG によって策定された仕様であり、2002 年に W3C 勧告となっている[2]。

XML 暗号では、既存の暗号技術を利用し、暗号化に関する情報（暗号化した結果のデータや復号に必要となる鍵の情報など、以下「暗号情報」と書く）を XML 形式で記述する方法を規定している。XML 暗号では、任意の形式のデータに対する暗号化をサポートしているが、XML データに対する暗号化の際に利用されることが多い。XML データに対する暗号化に XML 暗号を利用する場合、次のような利点がある。

- XML データの一部分に対する暗号化（部分暗号）が可能である。部分暗号を利用することで、例えば、XML データ内のうち秘匿性が求められる部分だけを暗号化するといった、高度な暗号処理が実現できる。
- 暗号対象のデータや暗号情報が全て XML 形式であるため、処理が行いやすい。例えば、暗号化されたデータと復号に必要となる鍵情報とを 1 つの XML データ内に格納するといったことが容易である。

2.3. 電子署名（XML 署名）

XML 署名（XML Signature Syntax and Processing）は、W3C と IETF の合同ワーキンググループである XML Signature WG によって策定された仕様である。XML 署名は 2002 年に W3C 勧告および RFC3275 となり、2008 年には XML 署名(Second Edition)が W3C 勧告となっている[3]。

XML 署名では、既存の署名技術を利用し、署名に関する情報（署名した結果である署名値や署名検証に必要となる鍵の情報など、以下「署名情報」と書く）を XML 形式で記述する方法を規定している。XML 署名は、任意の形式のデータに対する署名をサポートしているが、XML データに対する署名の際に用いられることが多い。XML データに対する署名に XML 署名を利用する場合、次のような利点がある。

- XML データの一部分に対する署名（部分署名）が可能である。部分署名を利用することで、例えば、XML データ内のうち、変更される可能性のある部分は署名対象から外し、変更しない（してはいけない）部分にのみ署名を行うといった、高度な署名処理が実現できる。
- 署名対象のデータと署名情報のデータが共に XML 形式であるため、処理が行いやすい。例えば、署名対象の XML データと署名情報とを 1 つの XML データ内に格納するといったことが容易である。

2.4. SOAP への適用（WS-Security）

XML を用いたシステム連携の標準プロトコルとして SOAP（Simple Object Access Protocol）がある。SOAP によるシステム連携において、システム間で交換する XML データを SOAP メッセージと呼ぶ。WS-Security（Web Services Security）は、SOAP メッセージのセキュリティを確保するための仕様である。WS-Security は、OASIS の WSS TC（Web Services Security Technical Committee）によって策定され、v1.0 が 2004 年に、v1.1 が 2006 年に OASIS 標準となっている[4]。

WS-Security では、SOAP メッセージのセキュリティを確保するため、主に次の 3 つの機能を提供している。

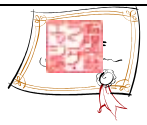
- XML 暗号を利用した SOAP メッセージの秘匿性の確保

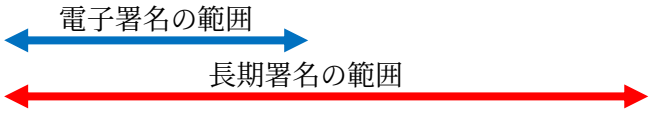
- XML 署名を利用した SOAP メッセージの完全性の確保
- SOAP メッセージへのセキュリティトークン（認証情報などのセキュリティに関する情報）の付与

2.5. 長期署名（XAdES）

XML署名について2.3節で紹介したが、この他に長期保存に対応するための長期署名仕様も存在する。 XAdESはXML署名に長期保存や他の高度な電子署名用の機能を加えた規格で、ヨーロッパのETSI （ www.etsi.org ）により、V1.3.2 が 2006 年 3 月に標準化されている。

2.5.1. 現実世界と電子世界の比較

現実世界	 印鑑（ハンコ）	 紙に押印した印影	 印鑑証明書や住民票等
電子世界	証明書と秘密鍵	電子署名（XML 署名）	検証情報（失効情報等）



2.5.2. 現実世界と電子世界の差異

1. **電子署名（証明書）は有効期限がある （現実より劣っている）**
   → 暗号方式の脆弱化の恐れ等から通常は数年から5年程度で無効になる。
   → **長期署名**は期限切れの前に更新する仕組み（仕様）で**期限延長**が可能。
2. **「誰が」だけで無く「いつ」もタイムスタンプ技術で保証可能 （現実より優れている）**
   → タイムスタンプは正しい時刻を保証する電子署名の一種でタイムスタンプ局が運営。
   → **長期署名**の仕様により**電子署名＋タイムスタンプ**を実現。

2.5.3. XML 署名と長期署名の比較

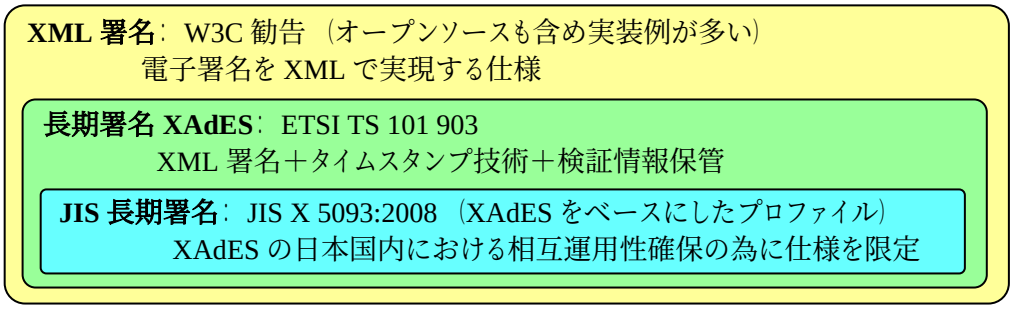


図 4 各種電子署名標準規格の関係

→ **短期的**に改竄や否認を防止する目的であれば **XML 署名**で良い。

→ 正確な**存在時刻を保証**するなら**長期署名 XAdES（XAdES-T）**が必要。

→ 有効期限をこえて**長期間保管**するなら**長期署名 XAdES （XAdES-A）**が必要。

2.5.4. 長期署名の課題

･法的対応の為には電子証明書や認証局等の **PKI の知識**や**利用の費用**が必要。

→ 逆に言えば法的対応しない（エンタープライズ PKI）**単なる XML 署名なら不要**。

･電子署名/タイムスタンプ仕様が**監督省庁や法律毎に異なっており**個別対応が必要。

→ **日本固有の問題**。

3. XML メッセージの保護の実践

3.1. オープンソースのツールによる暗号化と電子署名

2章で述べた XML 暗号、XML 署名は、商用製品やオープンソースなど様々なソフトウェアにおいて実装されている。本節では、代表的なオープンソース実装である Apache XML Security を用いて XML 暗号、XML 署名を実践する方法を述べる。

3.1.1. Apache XML Security の概要

Apache XML Security は、Apache Software Foundation の Apache XML プロジェクトによって開発されている XML 暗号･XML 署名の実装である。Apache XML Security は、当プロジェクトの Web サイト(http://xml.apache.org/security/)により Java および C++の実装が公開されている。以下では、Java 版の実装について述べる。

Apache XML Security では次の機能は実装しておらず、他の実装を使って実行することを前提としている。

- 暗号･署名エンジン – 2章でも述べたように、XML 暗号･XML 署名は、既存の暗号･署名技術を利用している。Apache XML Security では、XML 暗号･XML 署名で規定された処理は実装しているが、暗号･署名処理自体は実装していない。そのため、別途、暗号･署名エンジンが必要である。
- 鍵･証明書の管理 – Apache XML Security では、鍵･証明書の管理機能は提供していない。そのため、別途、鍵･証明書管理を行う実装が必要である。
- XML パーサ – Apache XML Security では、DOM (Document Object Model) によりパースされた XML データに対して暗号･署名を行うことができる。そのため、別途、DOM に対応した XML パーサが必要である。

なお、これらの機能はいずれも J2SE に標準で含まれている。そのため、J2SE に含まれている実装を使うのであれば、別途入手する必要はない。

3.1.3項以降では、Apache XML Security を用いて XML 暗号･XML 署名を実践する際のコード例を説明する。XML 暗号･XML 署名を実践するための Java の標準 API は、それぞれ JSR 106 (XML Digital Encryption APIs) と JSR 105 (XML Digital Signature APIs) で規定されている。JSR105 についてはすでに最終版の仕様が公開されており Java SE 6 にもその実装が取り込まれているが、JSR106 はまだ仕様策定途中である。そのため、本資料ではこれらの Java の標準 API ではなく、Apache XML Security の独自 API を用いたコード例について説明する。

3.1.2. システム構成

Apache XML Security を用いて、APSOM が担当するプランナから生産管理サーバに送

信する生産オーダ情報を保護する場合のシステム構成の例を図 5に示す。ここでは、生産管理サーバとスケジューラの両方で Apache XML Security を使用する場合の例を示しているが、どちらか片方のシステムを他の実装を用いて実現してもよい。

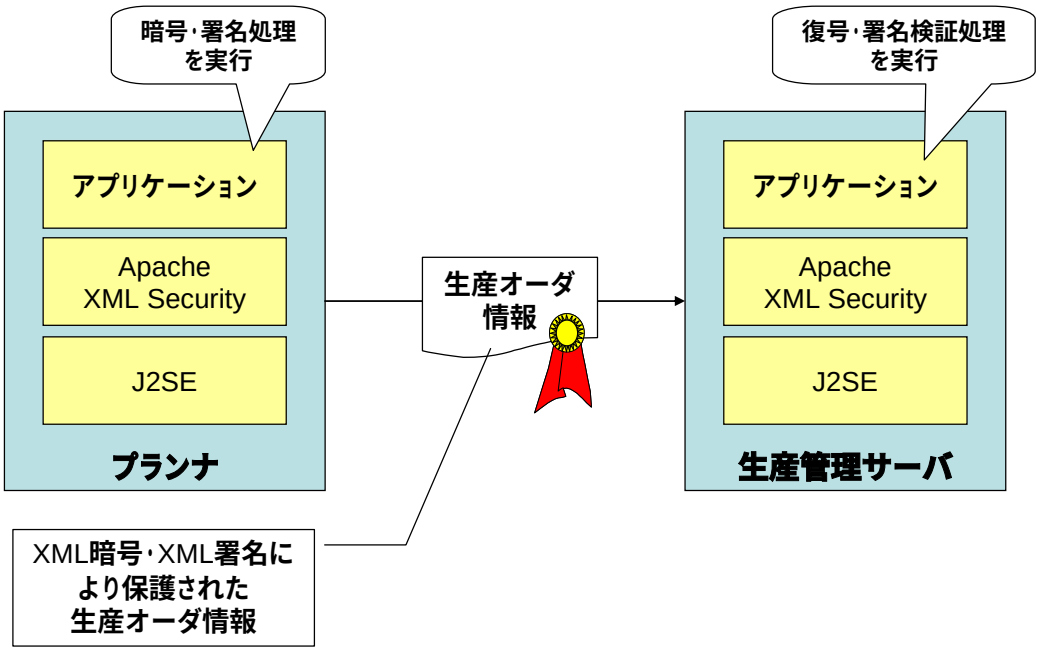


図 5 システム構成の例（Apache XML Security を使う場合）

3.1.3. XML 暗号の実践

Apache XML Security を用いた XML 暗号化のコード例をリスト 1 に示す。

```
1   // AES による私有鍵の生成
2   KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
3   keyGenerator.init(128);
4   Key symmetricKey = keyGenerator.generateKey();
5   // 私有鍵を暗号化する DES による共通鍵の生成
6   keyGenerator = KeyGenerator.getInstance("DESede");
7   SecretKey kek = keyGenerator.generateKey();
8   String algorithmURI = XMLCipher.TRIPLEDES_KeyWrap;
9   XMLCipher keyCipher = XMLCipher.getInstance(algorithmURI);
10  keyCipher.init(XMLCipher.WRAP_MODE, kek);
11  //共通鍵の暗号化
12  EncryptedKey encryptedKey =
```

```
13        keyCipher.encryptKey(document, symmetricKey);
14      // 暗号化方法の設定
15      XMLCipher xmlCipher = XMLCipher.getInstance(algorithmURI);
16      xmlCipher.init(XMLCipher.ENCRYPT_MODE, symmetricKey);
17      // 暗号化データに組み込む<KeyInfo>要素の設定
18      EncryptedData encryptedData = xmlCipher.getEncryptedData();
19      KeyInfo keyInfo = new KeyInfo(document);
20      keyInfo.add(encryptedKey);
21      encryptedData.setKeyInfo(keyInfo);
22      // 暗号化する要素名
23      Element enElement =
24       (Element) document.getElementsByTagName("要素名").item(int 要素位置);
25       // 暗号化すべきデータを EncrypteｄData 要素により置換
26       xmlCipher.doFinal(document, enElement, true);
```

リスト 1 Apache XML Security による XML 暗号化の実装

ここでｄocument は、暗号化対象の XML 文書を DOM によりパースした結果のドキュメント（org.w3c.dom.Document インターフェースの実装クラスのインスタンス）である。
2～4 行目では AES による秘密鍵を生成している。6～10 行目では秘密鍵を暗号化する DES による交換鍵を生成している。12～16 行目では共通鍵の暗号化と暗号化方法の設定を行っている。18～21 行目では復号に必要な暗号化データに組み込む<KeyInfo>要素の設定を行っている。23～24 行目では暗号化する要素名を設定する為に要素を特定している。26 行目では暗号化すべきデータを EncrypteｄData 要素で置換してｄocument に暗号化 XML 文書を完成させる doFinal メソッドを呼び出している。
リスト 1 には含まれていないが、最後に、暗号化済みのｄocument と共通鍵の kek をファイルに書き出す必要がある。

3.1.4. XML 署名の実践

Apache XML Security を用いた XML 署名のコード例をリスト 2 に示す。ここでは、Enveloped 署名と呼ばれる方式を用いて XML 文書全体に対して署名を行っている。

```
1       // Signature 要素の生成
2       XMLSignature sig = new XMLSignature(doc, null,
3                                   XMLSignature.ALGO_ID_SIGNATURE_DSA);
```

```
4    doc.getDocumentElement.appendChild(sig.getElement());
5    // Transforms 要素の生成
6    Transforms transforms = new Transforms(doc);
7    transforms.addTransform(Transforms.TRANSFORM_ENVELOPED_SIGNATURE);
8    transforms.addTransform(Transforms.TRANSFORM_C14N_WITH_COMMENTS);
9    // 署名対象情報の追加（Reference 要素の生成）
10   sig.addDocument("", transforms,
11             XMLSignature.ALGO_ID_SIGNATURE_RSA_SHA1);
12   // 鍵情報の追加（KeyInfo 要素の生成）
13   sig.addKeyInfo(cert);
14   // 署名を実行
15   sig.sign(privateKey);
```

リスト 2 Apache XML Security による XML 署名の実装

ここで、doc は署名対象の XML 文書を DOM によりパースした結果のドキュメント（org.w3c.dom.Document インターフェースの実装クラスのインスタンス）である。また、privateKey は署名の際に使用する私有鍵（java.security.PrivateKey クラスのインスタンス）であり、cert は署名検証に使用する X.509 証明書（java.security.cert.X509Certificate クラスのインスタンス）である。

2～3 行目では署名情報を表す Signature 要素を生成している。6～11 行目では、XML 署名の実行に必要となる各種アルゴリズムの設定や署名対象の設定を行っている。13 行目では、復号に必要となる鍵の情報を Signature 要素内に追加している。15 行目では、上記で設定した情報に基づき XML 署名を実行している。

3.2. ハードウェアを使った暗号化と電子署名

XML のセキュリティ機能は、ソフトウェアの形態だけでなく、ハードウェアとして実装・提供されている場合もある。 本節ではオープンソース実装の場合と同様に、APSOM が担当するプランナから送られる生産オーダ情報を、ハードウェアで XML 暗号、XML 署名を実践する方法を述べる。 ここではハードウェア形態のツールの例として日本セーフネット株式会社より提供を受けて LinaXML ハードウェアアプライアンスを使用した。

3.2.1. LunaXML の概要

LunaXML は XML 暗号、署名で使用する鍵を安全に管理し、暗号演算をハードウェア内部で実行できる。

OS は Linux を採用しており、鍵管理をハードウェアセキュリティモジュール(HSM)で行っている。XML 暗号、XML 署名といった標準機能を実装しており、Web サービスインターフェース（WSDL）、サンプルアプリケーション（Java）がすでに用意されているため、短期間で既存のシステムに対する導入が可能である。

以下に LunaXML の外観と仕様を示す。

図 6 LunaXML の外観

表 1 LunaXML の主な仕様

OS	Linux
インターフェース	SOAP/HTTPS
サポート暗号アルゴリズム	AES、TDES、RSA、ECDSA
XML セキュリティ	XML 暗号、XML 署名
エンコーディング	Base64
パフォーマンス	RSA1024 ビット鍵を使用した XML 署名は 1000 回/秒
セキュリティ	FIPS140-2 L3 認定のハードウェアセキュリティモジュール(HSM)を使用
ハードウェア仕様	GB イーサネットポート、1U フルレングス

- 暗号･署名エンジン

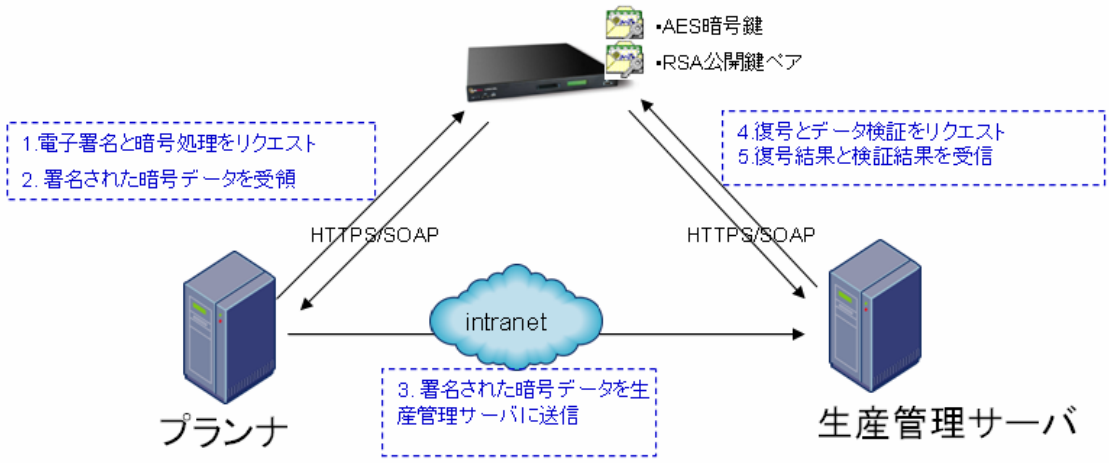
ハードウェア内に暗号･署名エンジンを内蔵している。 アクセラレーターにより大量の要求も高速に処理できる。

- 鍵･証明書の管理
  暗号鍵を安全に保管するための HSM を内蔵している。
- 利用形態
  SOAP/HTTPS でアクセス可能であり、Web サービスインターフェース（WSDL）が用意されている。 従ってアプリケーションのプラットフォームや言語に依存せず利用できる。

3.2.2. システム実装例

LunaXML を APSOM システムに導入することで以下のセキュリティ要件を満たすことが可能である。

- 生産オーダの機密性：LunaXML 標準機能の XML 暗号を使用することで実現
- 生産オーダの完全性：LunaXML 標準機能の XML 署名を使用することで実現



3.2.3. 検証結果

LunaXML 単体の機能を使用して、サンプルデータの暗号および署名処理を実行した。クライアントアプリケーションは日本セーフネットが用意しているサンプルアプリケーションを使用した。

システム環境を以下に示す。

- LunaXML：ファームウェアバージョン 4.6.1
- クライアント：WindowsXPSp2、JDK1.6、サンプルアプリケーション
- 対象データ：GanttTest.xml

結果としては、暗号、復号、署名および検証とも正常終了した。

また利用環境として、SOAP を使った Web サービスインターフェースによって XML デー

タの処理を要求し、結果を受け取れることを確認した。
今回使用した Web サービスインターフェースのいくつかを以下に記す。

表 2 LunaXML で使用したインターフェース

ログイン	<login> <UserID>user id<UserID> <authModel /> <password>password<password> <pkAuthentication>public key<pkAuthentication> </login>
XML 暗号	<xmlEncrypt> <Document ID="12345" RefURI="uri"> <EscapedXML>escaped XML</EscapedXML> <InlineXML>Some XML</InlineXML> <Base64XML>base64 encoded xml </Base64XML> <Base64Data>base64 data</Base64Data> </Document> <KeyInfo /> <AuthToken /> </ xmlEncrypt>
XML 復号	<xmlDecrypt> <Document> <InlineXML> <EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"> <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledescbc"/> <KeyInfo /> <CipherData> <CipherValue>base 64 value</CipherValue> </CipherData> </EncryptedData> </InlineXML> </Document> <AuthToken /> </xmlDecrypt>
XML 署名	<xmlSign Profile="urn:oasis:names:tc:dss:1.0:profile:dss_interop" RequestID="id"> <OptionalInputs> <KeySelector> <KeyInfo /> </KeySelector> <IncludeObject WhichDocument="12345" ObjectID="54321"/> </OptionalInputs> <InputDocuments> <Document ID="12345" RefURI="uri"> <EscapedXML>escaped XML</EscapedXML> <InlineXML>Some XML</InlineXML>

	<Base64XML>base64 encoded xml </Base64XML> <Base64Data>base64 data</Base64Data> </Document> </InputDocuments> <AuthToken /> </xmlSign>
XML 検証	<xmlVerify Profile="urn:oasis:names:tc:dss:1.0:profile:dss_interop" RequestID="2"> <OptionalInputs> <KeySelector> <KeyInfo /> </KeySelector> </OptionalInputs> <SignatureObject> <Signature> <SignedInfo> <CanonicalizationMethod Algorithm="algorithm uri"/> <SignatureMethod Algorithm="algorithm uri "/> <Reference URI="#54321"> <Transforms> <Transform Algorithm="algorithm uri”/> </Transforms> <DigestMethod Algorithm=”algorithm uri “/> <DigestValue>Base 64 Digest Value</DigestValue> </Reference> </SignedInfo> <SignatureValue>Base 64 Signature</SignatureValue> <Object Id=”54321”>document</Object> </Signature> </SignatureObject> <AuthToken /> </ xmlVerify>

3.3. 長期署名による XML データの保管

XML データを長期間にわたり保管する場合に、その内容の改竄防止や有効性を保証する為に XML 形式の長期署名 XAdES フォーマットを利用することができる。XAdES 長期署名も海外においては一部オープンソースプロジェクト（OpenXAdES）があり、日本においても商用製品が幾つか提供されている。本節では JIS 仕様である「JIS X 5093:2008 XML 署名利用電子署名(XAdES)の長期署名プロファイル」に適合した商用製品の例として Le-XAdES Library を用いて長期署名 XAdES を実践する方法を述べる。

3.3.1. Le-XAdES Library の概要

Le-XAdES Library は有限会社ラング・エッジにて開発されている XML 長期署名 XAdES の実装である。商用製品ではあるがソース公開がされておりテスト目的であれば自由に利用することができる。Web サイト（http://www.langedge.jp/pub/LeXAdES/）にて.NET を利用した実装がソース公開されている。将来的には Java による実装も計画されている。

3.3.2. 長期署名 XAdES の前準備

長期署名をおこなうには、署名に利用する（対応した私有鍵を有する）証明書とタイムスタンプサーバが必要となる。日本国内で法的に有効にする為にはどちらも有償のサービスを利用する必要があるが、ここではテスト目的の為に無償で提供されているサービスを紹介する。

- 証明書：ベリサイン社の無料テスト用 Class1 Digital ID（http://www.verisign.co.jp/welcome/try_verisign.html#class1） 60 日間の期間制限はあるが無料で取得できる。
- タイムスタンプ：アマノタイムスタンプビジネス社の AMANO Time Stamp Service Type-Free-A（http://www.e-timing.ne.jp/open_server/timestamp.html）主にテスト目的で公開されているタイムスタンプサーバ。

実運用する場合には、証明書は特定の種類のものを入手して使用する必要がある場合がある。 これは対象データ種別の監督省庁や適用法令によって規定され、有償（数千円～数万円）の場合もある。またタイムスタンプも財団法人日本データ通信協会の認定（http://www.dekyo.or.jp/tb/list/index.html）を受けた AMANO・PFU・セイコープレシジョン等の有償サービスを利用する必要がある。実運用での利用を検討する場合には各サービス提供元や専門業者と相談することを推奨する。

3.3.3. 長期署名 XAdES の実践

Le-XAdES Library を用いた長期署名 XAdES のコード例をリスト 1 に示す。ここでは

Enveloping 署名と呼ばれる方式を用いて XML 文書を含んだ長期署名を行っている。

```
1   // Le-XAdES Library の生成
2   le::ILeXAdES^ xades = le::makeLeXAdES();
3   // Enveloping(内包)形式で XML 文字列を追加
4   int rc = xades->addEnvelopingXml(xmlString);
5   // 証明書の選択
6   CAPICOM::Certificate^ cert = le::certSelect("証明書選択");
7   // 長期署名(XAdES-BES)の実施
8   rc = xades->sign(cert, true, SIGN_NO_SIGNTIME);
9   // 署名タイムスタンプ用ハッシュ値取得
10  ArrayList^ id = gcnew ArrayList();
11  array<Byte>^ hash1 = xades->getEsTHash(le::HASH_SHA512, id);
12  std::vector<byte> tst1 = le::getTsp3161(le::getVector(hash1),
13      "http://free-tsu.e-timing.ne.jp/TSS/HttpTspServer");
14  // 署名タイムスタンプ付与(XAdES-T)
15  rc = xades->addEsT(le::getArray(tst1), static_cast<String^>(id[0]));
16  // 検証情報付与(XAdES-X-Long)
17  rc = xades->addEsXL(nullptr, nullptr, nullptr);
18  // 保管タイムスタンプ用ハッシュ値取得
19  ArrayList^ ids = gcnew ArrayList();
20  array<Byte>^ hash2 = xades->getEsAHash(
21  le::HASH_SHA512, ids, nullptr, -1);
22  std::vector<byte> tst2 = le::getTsp3161(le::getVector(hash2),
23      "http://free-tsu.e-timing.ne.jp/TSS/HttpTspServer");
24  // 保管タイムスタンプ付与(XAdES-A)
25  rc = xades->addEsA(le::getArray(tst2), ids, nullptr);
26  // 長期署名 XAdES 文字列取得
27  String^ xadesString = xades->getXml();
```

リスト 3 Le-XAdES Library による長期署名 XAdES の実装

ここでは署名対象となる XML データは文字列（xmlString）として渡される前提となっている。結果も文字列（xadesString）として長期署名済み XML データとして取得する。

4. まとめ

4.1. XML データを保護する手段

MOF2008 合同デモシステムのモジュール間で送受信されるデータに対して、XML 暗号および XML 署名を適用することで、漏洩や改竄の危険性を低減することができる。 それらの技術と、電子署名の期限切れの懸念に対応するための長期署名技術について解説した。

さらに実際にデモシステムで通信される XML データのサンプルを用いて、各技術の利用方法を検証し、報告した。 セキュリティ機能を提供するツールとしてオープンソースとハードウェアという異なる形態のものを利用できたので、それぞれ検証した。 いずれも XML 暗号と XML 署名の適用を確認し、ただし利用形態の違いにより求められる環境や開発内容が異なることを確認した。 主な違いについては4.2節で検討する。

長期署名についても具体的なツールを用いてサンプルデータに適用した。

4.2. ツールの種類による実装比較

今回調査･検証した二種類の XML 暗号および XML 署名のツールについて、比較した結果を示す。

表 3 ツールの比較

ツール	Apache XML Security	LunaXML
形態	オープンソース	ハードウェア･アプライアンス
暗号･署名エンジン	別途必要、ただし J2SE には含まれている	内蔵
処理性能	プラットフォームに依存	アクセラレーター内蔵
鍵･証明書の管理	別途必要、ただし J2SE には含まれている	セキュリティ強度の高い HSM を内蔵
アプリケーション	Java、C++	非依存 (SOAP で呼び出し、WSDL 提供)
費用	無償、ただし実装に別途工数 (コスト) が必要	ハードウェアの購入が必要
入手の容易性	インターネットからダウンロードして利用可能	調達、導入の手続きが必要

4.3. XML セキュリティのツール

XML のセキュリティについては、今回取り上げたものの他にも、多くのツールが有償、無償含めて提供されている。 その多くは、

- アプリケーション･サーバの一機能

- ライブラリ
- ゲートウェイあるいは ESB

の何れかの形態で提供される実装のための製品と、互換性や性能を試験するためのテストツールとなっている。 XML コンソーシアムでは、こうしたツールの状況についても調査しており、別途報告を行っている[5]。

4.4. ハードウェアセキュリティモジュール(HSM)について

今回調査対象としたハードウェア製品においては、鍵管理機能に特化した HSM を内蔵していた。 HSM は暗号鍵の管理機能を提供すると同時に、暗号鍵の漏洩を防ぐ特殊な仕組みを装備している[6]。 ソフトウェアで実装する場合にも、別途 HSM を用意して利用することもできる。

HSM は、次のような機能を提供し、暗号処理におけるセキュリティ強度を高めることができる。

- 耐タンパのハードウェア内で鍵管理をすることで、悪意のある第 3 者から鍵を保護
- 各鍵を集中管理することでオペレーションミス等による鍵データの漏洩を防御
- 鍵の運用に関するセキュリティ機能
  - 鍵を使用するためには、複数人の認証が必要
  - 2 要素認証
  - 鍵のライフサイクル管理

参考資料

1 XMLコンソーシアム セキュリティ部会, “製造情報連携フォーラムSCF2007 デモシステム向けセキュリティ検討報告書”, http://www.xmlconsortium.org/wg/sec/securiry-proposal-071110a.pdf

2 W3C, “XML Encryption Syntax and Processing”, W3C Recommendation 10 December 2002, http://www.w3.org/TR/xmlenc-core/

3 W3C, “XML Signature Syntax and Processing (Second Edition)”, W3C Recommendation 10 June 2008, http://www.w3.org/TR/xmldsig-core/

4 XMLコンソーシアム セキュリティ部会, “Web Services Security 1.0 日本語訳”, 2005 年 03 月 31 日, http://www.xmlconsortium.org/wg/sec/wss.html

5 XMLコンソーシアム セキュリティ部会, “Web Services Security 製品対応状況について”, 第 4 回 XMLコンソーシアムWeek, 2005 年 6 月 7 日, http://www.xmlconsortium.org/seminar/050607-10-W04/050607-prog.html

6 Jim Attridge, “An Overview of Hardware Security Modules”, January 14, 2002, http://www.sans.org/reading_room/whitepapers/vpns/757.php