



**Livre Blanc de sécurité SOA.
Sécurité des données. XML
Signature, XML Encryption. v.1.0**

Project Documentation

Table of Contents

1 Introduction	
1.1 Avant Propos	1
2 La Mise en Oeuvre	
2.1 Sécurité Niveau Données	3
2.1.1 XML Signature	4
2.1.2 XML Encryption	8
2.2 Les Solutions Open Source	11
3 Annexes	
3.1 Où trouver les spécifications?	12

1.1 Avant Propos

«Les services Web XML vont rouvrir 70% des chemins d'attaques fermés par les parefeu lors de la dernière décennie. Ils peuvent transporter virtuellement toutes les données utiles sur le port 80 et le pare-feu ne peut les arrêter.»

- Gartner Group , 2003

Les services Web apportent des bénéfices significatifs pour des applications basées sur l'Architecture Orientée Services, mais exposent des risques importants en terme de sécurité. Créer et gérer un environnement sécurisé pour les services web nécessite la manipulation et la maîtrise de spécifications et standards divers et variés ainsi que des technologies et logicielles et matérielles conséquentes.

Il convient d'étudier la mise en œuvre de la sécurité pour les Architectures Orientées Services (SOA) via les quatre volets suivants :

- **La sécurité niveau Transport** : pare-feu (firewall), VPN (Virtual Private Networks), authentification basique, non-répudiation et cryptage
- **La sécurité niveau Message** : Utilisation des jetons de sécurité afin de valider l'identité du consommateur du service ou du processus, utilisation des assertions d'autorisation pour valider l'accès au services.
- **Sécurité niveau application** : Sécuriser les composants appelés par les Web Services, EJBs, Servlets appelés via les services Web.
- **Sécurité niveau Données** : Cryptage et signature des messages afin de protéger les données stockées ou transmises.
- **Sécurité niveau Environnement** : Monitoring, logging et audit afin d'identifier les problèmes qui doivent être fixés et résolus et établir des communications sûres et fiables.

L'émergence des services web pose plusieurs problématiques dont celle de la sécurité des échanges de messages entre partenaires. Dans une architecture Orientée Services, les services web peuvent exposer des processus métier sensibles qui nécessitent un traitement particulier en terme de sécurité aux deux bouts du canal de communication. En plus, les services web sont des technologies récentes, ceci implique de nouvelles vulnérabilités et attaques ou menaces.

Les services web sont utilisés dans les cas suivants (la liste n'est certainement pas exhaustive) :

- Intégration de systèmes point-à-point
- Intégration d'applications entreprise
- Collaboration et partenariat Business
- E-Business
- Composition des processus métier
- Protection et ouverture des systèmes d'information
- Réduction des coûts du cycle de vie du développement et la maintenance des systèmes d'information

Les solutions de sécurité des services web doivent prendre en charge les concepts suivants :

- L'authentification
- L'autorisation
- La confidentialité
- L'intégrité
- La non-répudiation

En plus de ces concepts, le système de sécurité doit prendre en charge l'audit des actions et messages envoyés afin de tracer l'activité de la sécurité des web services.

Les utilisateurs des services web doivent être identifiés soit via un nom d'utilisateur combiné à un mot de passe soit via un certificat digital. Une fois l'utilisateur identifié, il doit posséder l'autorisation ou l'habilitation nécessaire afin d'effectuer le traitement qu'il a demandé. Toutes informations ou messages sensibles mis en jeu par le traitement doivent être confidentiels et ne doivent pas subir d'altération qui touche à son intégrité d'origine. Une fois le traitement exécuté, des mesures de non-répudiation doivent être mises en place afin d'éviter tout dénis des deux parts (consommateurs/fournisseur).

Les services web reposent sur des topologies applicatives diverses et variées telles que : l'Internet mobile, des passerelles, zones démilitarisées (DMZ), systèmes distribués La communication entre ces technologies s'effectue via des intermédiaires.

La sécurité n'était pas la priorité des organisations travaillant sur les spécifications de la stack WS. La sécurité au niveau de la couche de Transport n'étant pas suffisante, il demeurait un vide qui freinait l'adoption des services web. Heureusement, plusieurs propositions majeures ont été diffusées afin de combler ce vide dont **WS-Security** qui apporte un support globale de l'intégrité, de la confidentialité et l'authentification des messages et **SAML** qui définit un langage commun d'interopérabilité permettant de partager les informations liées à l'authentification et l'autorisation afin de faciliter la mise en place de fonctionnalités SSO et de permettre la délégation des droits. Il est intéressant à noter que d'autres propositions et spécifications tendent à émerger comme **Kerberos** , **XKMS** , **XACML** afin d'apporter un support complémentaire à la stack de sécurité.

La sécurité des messages est d'autant plus nécessaire si des intermédiaires sont présents dans une communication point-à-point. L'expéditeur d'origine et le destinataire final doivent établir des relations de confiance avec ces intermédiaires afin d'assurer la sécurité de bout en bout.

2.1 Sécurité Niveau Données

XML est un format riche en terme de sémantique, fournit une représentation structurée des données, se décrit en fichiers texte et présente la facilité d'être utilisé dans un contexte Web. Tout ceci permet à XML de devenir un standard incontournable dans les échanges de type e-Business entre une entreprise et ses partenaires. Ceci dit, cet échange ouvre une difficulté ou un challenge, celui de sécuriser les données de ces échanges. Cette sécurisation consiste en le cryptage des données véhiculées dans le document XML ainsi que leur signature digitale.

Prenons un exemple très parlant : Dans le cas où plusieurs partenaires participent à une transaction donnée, chaque partenaire veut garder les données de la transaction qui le concerne sécurisée et identifiable. Les standards usuels de sécurisation ne permettent pas de choisir une portion des données à sécuriser vu leur pauvreté syntaxique à ce niveau.

Les initiatives de sécurité qui permettent de répondre à ces besoins sont les suivants : **XML Signature** et **XML Encryption**. Tous deux sont standards. Les chapitres à venir traitent de ces standards ainsi que les différentes étapes d'utilisation.

XML Signature et **XML Encryption** sont des spécifications fondamentales pour la protection des données relatives à des services web. C'est parce que les spécifications relatives aux services Web sont basées sur XML que ces technologies sont indispensables et intéressantes en même temps. Par exemple vous pouvez crypter un document WSDL pour le protéger des accès qui ne sont pas autorisés ou le signer si vous voulez le protéger contre la manipulation.

Les spécifications SAML, XACML et XKMS ne sont pas relatives aux services Web directement, elles sont adaptables pour ces derniers. C'est le cas contraire de **XML Signature** et **XML Encryption**, d'où l'intérêt de ces technologies.

Il est fondamental d'utiliser ces technologies quand des données XML doivent être protégées en dehors des messages SOAP et quand les metadata des services Web doivent être protégés de tout accès non autorisé. Ws-Security utilise **XML Signature** et XML Encryption afin d'assurer la confidentialité et l'intégrité des messages SOAP, mais il ne décrit pas comment utiliser ces technologies pour les données en dehors du contexte SOAP et WSDL, par exemple dans le cas d'applications qui passent par des formats intermédiaires de documents XML.

Utilisée conjointement avec le cryptage XML, une signature XML garantit que les données reçues correspondent exactement aux données transmises

2.1.1 XML Signature

Les signatures XML sont des signatures digitales utilisées pour fournir les mécanismes d'authentification, d'intégrité des données et la non répudiation. Le dispositif le plus intéressant de XML Signature et qui lui confère une faculté de flexibilité est la capacité à signer des portions spécifiques du document XML. Cette flexibilité peut être utile pour assurer l'intégrité de certaines parties signées par plusieurs partenaires participants à une seule et même transaction.

XML Signature peut gérer plusieurs types de données, il peut transporter par exemple du code HTML signé, un contenu binaire (un fichier ZIP), du contenu XML ...

Le processus de validation de la signature XML est effectif quand l'objet signé indique la référence de l'objet d'origine. XML Signature permet de gérer ces références à plusieurs niveaux :

- Une URI,
- La même que la ressource d'origine,
- Enveloppée dans l'entête du document signé (la signature est le père),
- Enveloppe l'objet signé (La signature est un enfant).

La signature des documents XML est fortement liée au cryptage.

D'un concept similaire à celles des certificats de sécurité, les signatures XML servent à garantir que l'intégrité du contenu d'un document XML et d'assurer au destinataire que ce dernier n'a pas changé en cours de route. La signature des fichiers XML utilise une notion appelée « canonicalization » pour compenser les différences typographiques des systèmes de fichiers et parseurs correspondants.

Lorsqu'une signature est appliquée au contenu, la canonicalization utilise les données et balises du fichier XML pour créer une signature unique, ignorant des informations telles que les sauts de ligne et les tabulations.

Lors de la réception d'un document, le système client effectue une "transformation de la signature XML", qui fait la distinction entre le contenu crypté avant signature et le contenu crypté après. Toutes les données cryptées après la signature sont décryptées ; l'intégrité des données est vérifiée en appliquant la même méthode de canonicalization au contenu, par comparaison du résultat à la signature incluse dans le document XML.

Les composants de XML Signature

Balise	Description
Reference	Chaque ressource à signer est associée à sa propre référence identifiée via l'attribut URI.
Transforms	Cet élément spécifie la liste des étapes appliquées au contenu de la ressource référencée avant de subir un digest.
DigestValue	Cet élément comporte la valeur du digest correspondant à la ressource référencée.

SignatureValue	Cet élément spécifie la valeur du digest crypté de l'élément SignedInfo
KEyInfo	Cet élément indique la clef à utiliser pour valider la signature. Cette information peut renseigner aussi bien des certificats, des noms de clefs, des algorithmes d'arrangement (agreement keys)

Comment utiliser XML Signature ?

Ci-dessous un descriptif sommaire des étapes à suivre pour signer un document XML :

1. Déterminer la ressource à signer, la ressource à signer correspond à une URI selon le type de la ressource, par exemple :
 - <http://www.opencap.org/transaction-result.html> (référence à un document HTML)
 - <http://www.opencap.org/transation-result.jpeg> (référence à une image)
 - <http://www.opencap.org/transaction-result.xml> (référence au fichier transaction-result.xml)
 - <http://www.opencap.org/transaction-result.xml#total> (référence à l'élément nommé total dans le fichier transaction-result.xml)
2. Déterminer le digest de chaque ressource à signer.

Chaque ressource est associée à un élément <Reference> et dont le contenu signé est placé dans l'élément <DigestValue>, par exemple

```
<Reference
  URI="http://www.opencap.org/transaction-result.xml">
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
  <DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=</DigestValue>
</Reference>
```

<DigestMethod> spécifie l'algorithme utilisé pour calculer le digest.

3. Collecter les éléments <Reference>

Envelopper les références dans un élément <SignedInfo>, par exemple :

```
<SignedInfo Id="id">
  <CanonicalizationMethod
    Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
  <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1" />
  <Reference
    URI="http://www.opencap.org/transaction-result.xml">
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
    <DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=</DigestValue>
  </Reference>
</SignedInfo>
```

L'élément `<CanonicalizationMethod>` détermine l'algorithme utilisé pour canoniser l'élément `<SignedInfo>`. `<SignatureMethod>` identifie l'algorithme utilisé pour produire la valeur de la signature.

4. La signature

Calculer le digest de l'élément `<SignedInfo>` et le renseigner via la déclaration

```
<SignatureValue>VVVVVVVV=</SignatureValue>
```

5. Ajouter des informations

Inclure la clef publique pour la vérification de la signature, par exemple :

```
<KeyInfo>
  <X509Data>
    <X509SubjectName>CN=AALAMI,O=OpenCap.,ST=BDX,C=FR</X509SubjectName>
    <X509Certificate>MIID51VN</X509Certificate>
  </X509Data>
</KeyInfo>
```

6. Construire le document XML final:

```
<?xml version="1.0" encoding="UTF-8"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo Id="id">
    <CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1" />
    <Reference
      URI="http://www.opencap.org/transaction-result.xml">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>VVVVVVVV=</SignatureValue>
</KeyInfo>
  <X509Data>
    <X509SubjectName>CN=AALAMI,O=OpenCap.,ST=BDX,C=FR</X509SubjectName>
    <X509Certificate>MIID51VN</X509Certificate>
  </X509Data>
</KeyInfo>
</Signature>
```


2.1.2 XML Encryption

Les technologies SSL et TLS permettent de sécuriser les échanges point-to-point et ne permettent pas de définir un cryptage sélectif des données. Par exemple, plusieurs parties sont entrent en jeu dans le workflow d'un système de paiement en ligne, le payeur, l'organisme bancaire recevant l'ordre de débit, l'organisme bancaire recevant l'ordre de crédit et le fournisseur du produit, on voit donc les limitations des protocoles courants surtout quand il s'agit d'une transaction end-to-end comme dans ce cas là. Une autre limitation est celle que SSL crypte le contenu global des données transmises ce qui n'est pas commode dans ce type de transaction où les acteurs n'ont pas besoin de traduire quelques informations transmises.

XML Encryption répond à ces lacunes et s'insere dans une démarche de sécurisation des documents XML echangés via des services Web.

XML Encryption est aussi très utile lorsque plusieurs parties du même documents ont besoin d'un traitement différent, par exemple, le fournisseur du produit n'a pas besoin de connaître les détails sur la carte de crédit utilisée par la banque, et la banque n'a pas besoin de connaître les détails des produits achetés.

XML Encryption est utilisé afin de crypter des données qui peuvent être :

- Des données arbitraires,
- Un élément XML,
- Le contenu d'un élément XML.

Lors du cryptage d'un élément ou du contenu d'un élément XML, l'élément EncryptedData remplace l'élément ou le contenu (respectivement) dans la version cryptée du document XML.

Lors du cryptage de données arbitraires (y compris des documents XML entiers), l'élément EncryptedData peut devenir la racine d'un nouveau document XML ou devenir un élément enfant dans un document XML choisi par l'application.

L'élément <EncryptedData> est constitué de plusieurs sous-éléments contenant des informations sur les données cryptées, notamment sur les clés et le code de chiffrement réel (ou la référence à ce code).

XML Encryption définit une notion de granularité qui permet de décider des éléments du document à crypter ainsi que l'algorithme utilisé.

Ci-dessous la structure de l'élément EncryptedData :

```
<EncryptedData Id? Type?>
  <EncryptionMethod/?>
  <ds:KeyInfo>
    <EncryptedKey?>
    <AgreementMethod?>
    <ds:KeyName?>
```

```

    <ds:RetrievalMethod>?
    <ds:*>?
  </ds:KeyInfo>?
  <CipherData>
    <CipherValue>?
    <CipherReference URI?>?
  </CipherData>
  <EncryptionProperties>?
</EncryptedData>

```

Exemple

Voici par exemple le fichier XML transmis pour la transaction à la banque qui s'occupe de créditer le fournisseur. Tout passe en clair.

```

<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>

```

La première solution consiste à crypter le document en entier, ci-dessous le résultat.

```

<?xml version='1.0'?>
  <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
    Type='http://www.isi.edu/in-notes/iana/assignments/media-
    types/text/xml'>
    <CipherData><CipherValue>A23B45C56</CipherValue>
  </CipherData>
</EncryptedData>

```

XML Encryption propose aussi un sur-cryptage, ci-dessous un exemple résultat. Un document XML peut contenir zéro ou plus éléments EncryptedData. L'élément EncryptedData ne peut être le parent ou l'enfant d'un autre élément EncryptedData. Cependant, les données cryptées en question peuvent être n'importe quoi, y compris des éléments EncryptedData ou EncryptedKey

```

<pay:InfoPaiement xmlns:pay='http://exemple.org/paiementv2'>
  <EncryptedData Id='ED1' xmlns='http://www.w3.org/2001/04/xmlenc#'

```

```

Type='http://www.w3.org/2001/04/xmlenc#Element'>
  <CipherData>
    <CipherValue>EncryptedDataOriginal</CipherValue>
  </CipherData>
</EncryptedData>
</pay:InfoPaieement>

```

En utilisant XML Encryption, on a la possibilité de crypter les informations liées à la carte bancaire du client. Ci-dessous le résultat de cryptage, notons la valeur de l'attribut Type de la balise <EncryptedData> qui renseigne que le contenu crypté correspond à un élément XML.

```

<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
    xmlns='http://www.w3.org/2001/04/xmlenc#'>
    <CipherData><CipherValue>A23B45C56
    </CipherValue></CipherData>
  </EncryptedData>
</PaymentInfo>

```

La deuxième solution consiste à crypter que le numéro de la carte bancaire. Ci-dessous le résultat, notons la valeur de l'attribut Type de la balise <EncryptedData> qui renseigne que le contenu crypté correspond à un contenu d'un élément XML.

```

<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>
      <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
        Type='http://www.w3.org/2001/04/xmlenc#Content'>
        <CipherData><CipherValue>A23B45C56</CipherValue>
      </CipherData>
    </EncryptedData>
    </Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>

```

2.2 Les Solutions Open Source

Il existe plusieurs projets travaillant sur l'implémentation de spécifications relatives à la sécurité des services Web. La majorité de ces implémentations utilisent les langages Java ou C++. Ci-dessous un descriptif des projets les plus conséquents :

- Apache XML Security : Ce projet a pour but de fournir une implémentation des standards de sécurité relatifs à XML. Ces standards sont XML-Signature et XML Processing. Le projet vise de travailler sur les spécifications XML Key Management (XKMS)
- OpenSAML consiste en un ensemble de bibliothèques Open Source Java et C++ qui sont en conformité avec les spécifications SAML 1.0 et 1.1.
- Apache Directory Project : Ce projet livre un module de sécurité comportant :
 - Un framework d'authentification : AuthX
 - Une librairie compatible RFC Kerberos
 - ChangePw pour changer les mots de passe de manière sécurisée.
- VeriSign WS-Security toolkit : Librairie Open Source pour faciliter l'utilisation et l'intégration de WS-Security dans le développement des applications sécurisées basées sur les services Web.
- La Crypto API de Bouncy Castle inclut les points suivants:
 - Une API légère de cryptographie en Java.
 - Un fournisseur pour la JCE et JCA.
 - Une implémentation convenable de la JCE 1.2.1.
 - Une bibliothèque pour lire et écrire des objets encodés en ASN.1.
 - Des Générateurs pour les versions 1 et 3 des certificats X.509 et des fichiers PKCS12.
 - Des Générateurs pour la version 2 des certificats de l'attribut X.509.
 - Des Générateurs/Processeurs pour S/MIME et CMS (PKCS7).
 - Des Générateurs/Processeurs pour OCSP (RFC 2560).
 - Des Générateurs/Processeurs pour TSP (RFC 3161).
 - Des Générateurs/Processeurs pour OpenPGP (RFC 2440).
 - Une version jar signée, appropriée pour les JDK 1.4/1.5 et la JCE de Sun.

3.1 Où trouver les spécifications?

SAML	http://www.oasis-open.org/committees/security/
Security Services TC	http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
WS-Federation	http://www-106.ibm.com/developerworks/webservices/library/ws-fedworld/
WS-Security	http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
WS-SecureConversation	http://www-106.ibm.com/developerworks/webservices/library/ws-secon/
WS-SecurityPolicy	http://www-106.ibm.com/developerworks/webservices/library/ws-secpol/
WS-Trust	http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-trust.asp
XML-Encryption	http://www.w3c.org/Encryption/2001/
XML-Signature	http://www.w3c.org/Signature/