XML et Sécurité

Didier DONSEZ
Université Joseph Fourier
IMA –IMAG/LSR/ADELE
Didier.Donsez@imag.fr

Motivations : Échanges B2B et B2C

- Signature et Chiffrement
  de portions de sous-documents XML
  - Par un ou plusieurs signataires
- Sécurisation des enveloppes SOAP pour les Web Services
- Interopérabilité pour la gestion des clés (PKI)

Standards

- XML Signature
- XML Encryption
- XKMS - XML Key Management System
- SAML - Security Assertions Markup Language
- XACML - XML Access Control Markup Language
- EIEIO - proposed by McDonalds

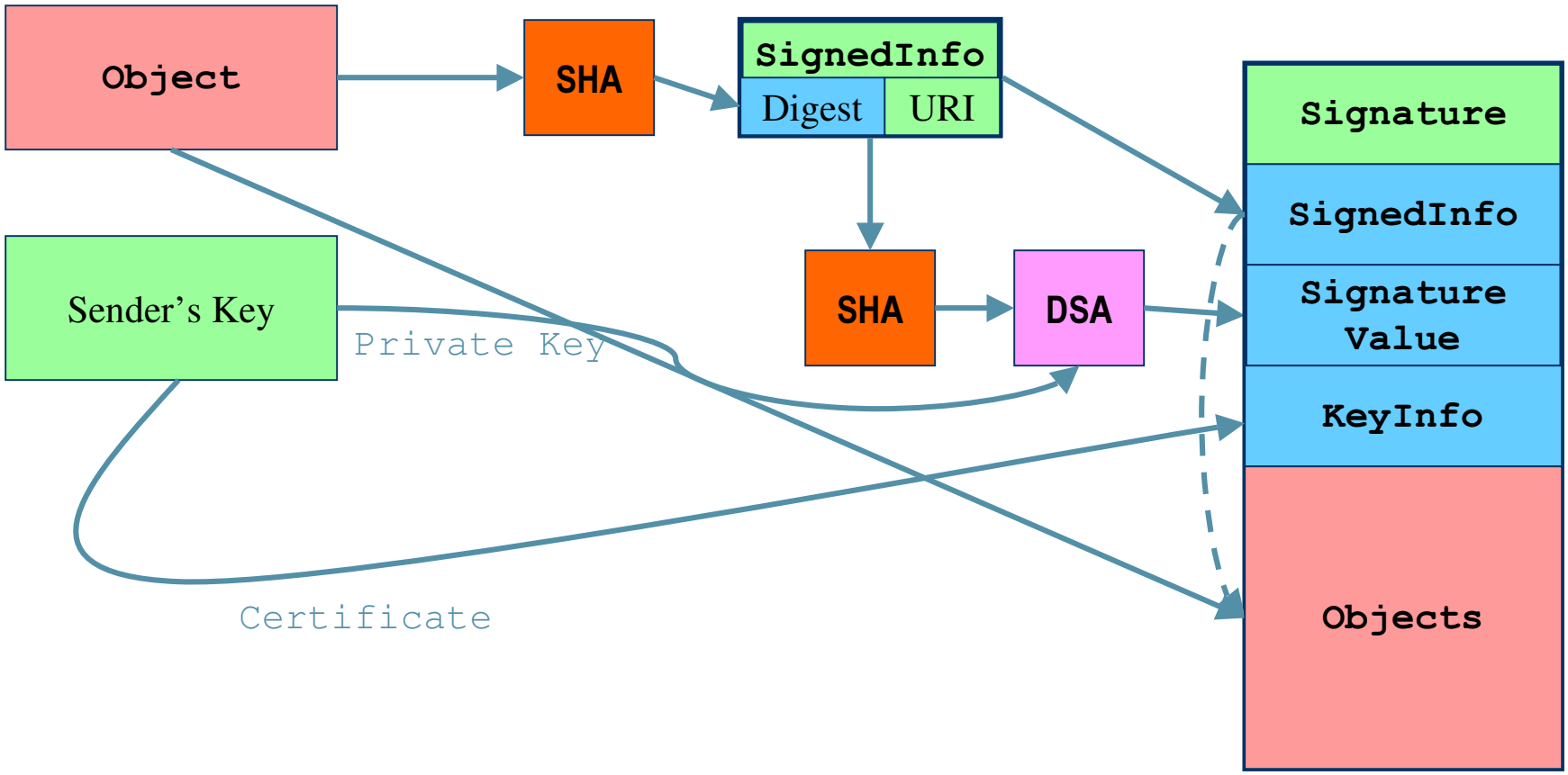
XML Signature

- Objectifs
  - Prévenir la falsification d'une message (intermédiaire, récepteur) ou sa répudiation (émetteur)
  - Le document doit pouvoir être partiellement signé pour continuer à être modifier sur d'autres parties
- Fonctionnalités
  - Basé sur XML (pas de notation ASN.1)
  - Signatures sur des portions (arbres d'éléments) de documents
    - Le document peut être recomposé/transformé par le récepteur
  - Signatures sur plusieurs entités
    - 1 document HTML + feuille CSS + image de bannière + …
  - Signatures par plusieurs signataires
    - Un billet combiné SNCF et Air France (2 signataires)
  - Signatures embarqués dans le document (signature dite enveloppante)
    - SOAP request/response
  - Signature d'entités externes référencées par des URI (signature détaché)
    - HTML, JPEG, MPEG …

XML Signature

- Traitement
  - L'émetteur crée un message, le canonise et le signe
  - Le récepteur recoit un message, le canonise et vérifie sa signature
- Canonisation
  - Indépendance de l'indentation (esp. Cr, ..)
    sur le calcul de la signature
- W3C et IETF
  - XML Digital Signature http://www.w3.org/TR/xmldsig-core/
  - API Java
    - JSR 105 XML Digital Signature APIs (javax.security.xml.dsig)

XML Signature
Construction du message signé



XML Signature
exemple de signature détachée

```
<?xml version="1.0"?>
<Signature Id="MyFirstSignature" xmlns=http://www.w3.org/2000/09/xmldsig#>
<SignedInfo>
 <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-20001026"/>
 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
 <Reference URI="http://www-adele.imag.fr/~donsez">
  <Transforms>
   <Transform Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-20001026"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
 </Reference>
</SignedInfo>
 <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
</Signature>
```

Le résumé porte sur ma home page canonisée

La signature porte sur l'élément SignedInfo

XML Signature Transforms

Motivation

- Liste la liste des transformations à procéder sur l'objet (l'URI) pour la signature
  - Permet de ne signer qu'une partie d'un document (et de ne pas signer les parties variables)

Transformations possibles

- Canonisation, Encodage/Décodage (Compression/Décompression), XSLT, Xpath, validation XML Schema, Xinclude, ..

XML Signature
Informations supplémentaires

- Élément <Object>
  - Attention : il est optionel et le récepteur peut ne pas le comprendre
  - Les sous éléments doivent être identifiés
- Sous Elément <SignatureProperties>
  - Permet d'ajouter des informations complémentaires à la signature
    - Date de validité de la signature, ident du processus hard qui à signer, …
- Sous Element <Manifest>
  - Permet de signer indépendamment chaque URI listé dans le manifeste

XML Signature
Informations supplémentaires

- Exemple

XML Signature
Elément <KeyInfo>

- Élément optionel
- Permet au récepteur d'obtenir la clé nécessaire à valider la signature
  - KeyName (un nom, un index dans un repertoire de clé, un X500 DN, …)
  - RetreivalMethod
  - Certificats : X509, PGP, SPKI
  - Clés publiques : RSA, DSA

XML Signature
Exemple d'application : Ticketing

- Examples
  - Event, Plane, Transportation Pass, , Lottery, Car Wash, Telephone Card, Digital, Cash, Software License, Loyalty Bonus, Transportation Pass, Gate Card, Driver's license, ID Credit note (*avoir*), Discount Card…
- Promise
  - Signed by the issuer(s) (*based on XML Sig*)
  - Signed by the owner (*based on XML Sig*)
  - Properties (*based on RDF and XML Schema*)
    - Control Parameters : TicketID, IssuerID, OwnerOD, Validity, View, …
    - Promise

      Industry-Specific : Fligh number, Seat Number, Class, Event Name, …

      Issuer specific : Mileage points, advertissement, …
  - PK and Certificates, …
  - Attached ticket/description
    - Transfert to Owner2, …

XML Signature
Exemple d'application : Form Signing

- Formulaire rempli par le client
- Signé par lui

XML Encryption

- Motivation: chiffrer des portions (élément) d'un doc. XML
  - Différents destinataires doivent pouvoir lire différentes parties du document
  - La sécurité doit être maintenue de bout en bout
    - Ce n'est pas le cas avec SSL ou S/MIME
  - Exemple: une place de marché (Trading hub) doit pouvoir voir les information de catégorisation d'une réponse à une offre sans voir les informations de prix destinés à l'acheteur
- W3C
  - XML Digital Encryption
    - http://www.w3.org/TR/xmlenc-core/
  - API Java
    - JSR 106 XML Digital Encryption APIs

XML Encryption exemple

```
<?xml version='1.0'?>
<employee id='3456'>
  <name>John Smith</name>
  <title>Senior Analyst</title>
  <salary>
    <xenc:EncryptedData xmlns:xenc='http://www.w3.org/2000/11/temp-xmlenc'
      Type='NodeList'>
      <xenc:CipherText>AbCd....wXYZ</xenc:CipherText>
    </xenc:EncryptedData>
  </salary>
</employee>
```

Remarque :
la validité de ce document ne peut pas être totalement vérifié par les intermédiaires

The contents of the <salary> element are replaced with the <EncryptedData> element.

XML Encryption exemple avec SOAP

Remarque :
la sécurité est assurée de bout en bout contrairement à SSL et S/MIME

```
<?xml version="1.0"?>
<soap:Envelope ...>
<soap:Header>
 <encx:Encryption ...><KeyInfo>.....</KeyInfo></encx:Encryption>
</soap:Header>
<soap:Body>
 <StockQuote>
  <encx:EncryptedData>k%$#989hskdf&</encx:EncryptedData>
 </StockQuote>
</soap:Body>
</soap:Envelope>
```

The necessary <KeyInfo> information for the <EncryptedData> is carried in the SOAP header.

Exemple d'utilisation

```
// Open a document we're going play with.
  XMLPlainText xmlPlainText = new XMLPlainText(new FileInputStream("payment.xml"));
  // Let's load the keystore.
  KeyStore ks = KeyStore.getInstance("jceks");
  ks.load(new FileInputStream("jwstore"), "jwpassword".toCharArray());
  // Tell the class how to get the keys. KeyStoreKeyInfoResolver is just one way to get the keys.
  KeyStoreKeyInfoResolver kskiResolver = new KeyStoreKeyInfoResolver(ks);
  kskiResolver.putAliasAndPassword("bank", "bankpassword".toCharArray());
  // Prepare the encrypted data,  put CipherValue in it.
  EncryptedData ed = new EncryptedData();
  ed.setCipherData(prepareCipherData());
  // Use RSA v1.5 to encrypt.
  EncryptionMethod rsa_1_5 = prepareEncryptionMethod(EncryptionMethod.RSA_1_5);
  // Prepare the keys.
  KeyInfo bank = prepareNameOnlyKeyInfo("bank");
  // Encrypt!
  xmlPlainText.encrypt("//CreditCard", kskiResolver, ed, EncryptedData.ELEMENT, rsa_1_5, bank);
  // Dump it!
  PrintWriter pw = new PrintWriter( new FileWriter("encryptedPayment_rsa_1.xml"));
  pw.write(xmlPlainText.toString());   pw.close();
```

Usually you'd use a 3DES key to encrypt, and the RSAkey to encrypt the 3DES key

```
<?xml version="1.0" encoding="UTF-8" ?>
<PurchaseOrderRequest>
  <Order>
    <Item>
      <Code>1234567890</Co
      <Description>Digital Cam
    </Item>
    <Quantity>1</Quantity>
  </Order>
  <Payment>
    <CreditCard>
      <BrandId>Brand X</Bran
      <Number>111122223333
      <ExpiryDate>20030408<
    </CreditCard>
    <PurchaseAmount>
      <Amount>123623</Amou
      <Currency>840</Currenc
    </PurchaseAmount>
  </Payment>
</PurchaseOrderRequest>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<PurchaseOrderRequest>
  <Order>
    <Item>
      <Code>1234567890</Code>
      <Description>Digital Camera</Description>
    </Item>
    <Quantity>1</Quantity>
  </Order>
  <Payment>
    <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#">
      <CipherData>
        <CipherValue>ci3q/7BqV92eizpxvIszQeo6GYB1NOfe
PoFiUvJtsYMkhrvR39cqUpdg07J3Q …
ihtoH55NZnjns78i4bNGlGoyNK4GTG9DTGELRTxA==</CipherValue>
      </CipherData>
    </EncryptedData>
    <PurchaseAmount>
      <Amount>123623</Amount>
      <Currency>840</Currency>
    </PurchaseAmount>
  </Payment>
</PurchaseOrderRequest>
```

XML Encryption
Une application : les droits

- Exprimer/Contrôler les droits
  - rights=grants/licenses

- Format
  - XrML - eXtensible rights Markup Language
    - http://www.xrml.org
    - RDF, XML Signature, XML Encryption
    - Se base sur les spécifications MPEG-21 et TV-AnyTime
  - ODRL - Open Digital Rights Language
    - http://www.odrl.net
  - XCML - Extensible Media Commerce Language

XKMS - XML Key Management System

- Motivation
  - Replacer les formats et protocoles PKI (PKIX, Card Management Services, OCSP, etc.) par des documents XML transportés par SOAP.
- Définit les messages de requête et de réponse pour
  - Requérir (request) un certificat
  - Renouveler (renew) un certificat
  - Valider (validate) un certificat (expiration, CRL, OCSP, etc.)
  - Révoquer (revoke) un certificat (CRL)
- Basé sur XML Signature & XML Encryption
- W3C
  - Initié
  - XKMS XML Key Management Specification
    - http://www.xmltrustcenter.org/xkms
  - API Java
    - JSR 104 XML Trust Service APIs

XKMS
Exemple de message de révocation

A request to revoke the key specified by <KeyID>

```
<?xml version="1.0"?>
<Request>
    <Prototype>
        <AssertionStatus>Invalid</AssertionStatus>
        <KeyID>unique_key_identifier</KeyID>
        <ds:KeyInfo>……</ds:KeyInfo>
    </Prototype>
    <AuthInfo>
        <AuthUserInfo>
            <ProofOfPossession>[RSA-Sign]</ProofOfPossesion>
        </AuthUserInfo>
    </AuthInfo>
    <Respond>
        <string>KeyName</string>
    </Respond>
</Request>
```

SAML Security Assertions Markup Language

But:

- Representer l'authentification et les décisions d'autorisation (en XML) pour les Web Services, l'EAI
  - Who/What/When/Why/Where and How?
- Les applications demandent l'authentification ou l'autorisation auprès de serveurs de securité.
- Permettre l'interoperabilité entre les serveurs de sécurité comme Netegrity SiteMinder, Securant ClearTrust, Oblix NetPoint, IBM PolicyMaker, etc.

Histoire

- S2ML (Security Services Markup Language)
  - Netegrity, Sun, webMethods, Verisign, etc.
- AuthXML (Authorization/Authentication XML)
  - Securant & Outlook Technologies
- Fusion sous la bannière d'OASIS Security Services Technical Committee (SSTC) http://www.oasis-open.org/committees/security/index.shtml

SAML Exemple

This entitlement describes the credit rating for a previously authenticated party.

The issuer signs the entitlement to bind its identity to the entitlement assertion.

```
<?xml version="1.0"?>
<Entitlement>
    <Id>urn:988876</Id>
    <Issuer>ExchangeA</Issuer>
    <Date>2000:12:23:12.00</Date>
    <Audiences>all</Audiences>
    <DependsOn>id4558999</DependsOn>
    <AzData>
        <cr:CreditRating>AAA+</cr:CreditRating>
    </AzData>
    <dsig:Signature>j&6fhl$3kppsdf</dsig:Signature>
</Entitlement>
```

XACML XML Access Control Markup Language

- Motivation
  - Représenter les politiques de contrôle d'accès (Access Control) en XML
- Norme
  - Proposition OASIS en cours

XACML Exemple

```
<?xml version="1.0"?>
<!-- Alice can read the creditcard element, but cannot modify  its contents. -->
<policy>
    <xacl>
        <object href="purchaseorder/creditcardnumber"/> <!– Xpath expr. -->
        <rule>
            <acl>
                <subject>
                    <uid>Alice</uid>
                </subject>
                <action name="read" permission="grant"/>
                <action name="write" permission="deny"/>
            </acl>
        </rule>
    </xacl>
</policy>
```