XML
Emerging Authorization and Authentication Standards

About the speaker

- Name: Mohammad al-Kahtani
- Doctoral Candidate at GMU
- Educational Background:
  - BS in Computer Science:
    Riyadh University, Saudi Arabia
  - Masters in Software Engineering:
    George Mason University
- Contact info: malkahta@gmu.edu
Topics of the Presentation

1. XML

2. Brief description for main emerging protocols
   a. SAML
   b. XACML
   c. XKMS
   d. XML signature

XML
(Extensible Markup Language)
**XML in brief**

- XML is **subset** of the Standard Generalized Markup Language (SGML)

- It is designed to make it easy to interchange structured documents over the Internet.

- Structured documents contain:
  1. Content (words, pictures, etc.) and
  2. Some indication of what role that content plays

- A markup language is a mechanism to identify structures in a document.

**XML and HTML**

- In HTML, both the *tag semantics and the tag set are fixed*:
  - New changes are confined by:
    1. Browser implementations
    2. Backward compatibility is paramount.

- XML specifies **neither semantics nor a tag set**.

- XML is really a meta-language for describing markup languages:
  - Provides a facility to define tags and the structural relationships between them.
  - No predefined tag set means there can't be any preconceived semantics.
  - All of the semantics of an XML document will either be defined by the applications that process them or by stylesheets.
Why XML?

- XML was created so that richly structured documents could be used over the web.

- The only viable alternatives, HTML and SGML, are not practical for this purpose:

  1. HTML comes bound with a set of semantics and does not provide arbitrary structure.

  2. SGML provides arbitrary structure, but is too difficult to implement just for a web browser.

An XML example

```xml
<?xml version="1.0"?>
<Country>
  <Summary>
    <Geographical_Location>
      Arabian Peninsula
    </Geographical_Location>
    <Population>
      14 Million
    </Population>
    <Religion>
      Islam
    </Religion>
  </Summary>
  <body>
    ***************
  </body>
</Country>
```
Main XML security protocols

SAML

assertions

XACML

assertions

User$_i$

credentials

User$_j$

credentials

User$_i$/principal

assertions

User$_j$/principal

assertions

encrypted/signed XML doc

public key

"credentials"

"credentials"

XML signature

XML encryption

XKMS

Service Provider

Resource

Main XML security protocols

SAML

(Secure Assertion Markup Language)
What is SAML

- A proposed standard for the exchange of **authentication** and **authorization** information between trust domains.

- SAML enables Single Sign On across trust domains
## SAML Assertions

The **basic data objects** of the SAML protocol model are "**Assertions**" and "**References**" (to Assertions).

1. **Authentication** Assertion: Asserts that the issuer has authenticated the specified subject

2. **Attribute** Assertion: Asserts that the specified subject has the specified attribute(s).

3. **Authorization** Assertion: Asserts that a subject has been granted specific permissions to access one or more resources.

## SAML

SAML Assertions may be exchanged using a variety of protocols:

1. The request protocol
   (defined by the `<SAMLQuery>` and `<SAMLQueryResponse>` elements)
2. HTTP
3. SMTP
4. MIME
5. ebXML
6. SOAP/XP
7. BEEP
SAML Static Domain Model

SAML: Scenario #1: Single Sign-On (Pull Model)
SAML: Scenario #1: Single Sign-On (Push Model)

Nov. 15, 2001 Mohammad al-Kahtani

SAML: Scenario #2: Authorization Service

Nov. 15, 2001 Mohammad al-Kahtani
The different types of SAML assertion are encoded in a common XML package, which consists of:

Basic Information:

- A unique identifier: Serves as a name for the assertion.
- SAML version no.
- Date and time of issue: Optional
- Time interval for which the assertion is valid: Optional

Describe the use of assertions to make claims for Authorization

1. “DecisionClaim”: Access permissions specified in the request identified by the corresponding RequestID were either permitted, denied or could not be determined

2. “AuthenticationClaim”: Specified subject has been authenticated

3. “AttributeClaim” element: Specified subject has the specified attribute(s) specified by a URI

4. “AuthorizationClaim”: Specified subject is authorized to perform the specified operation(s) on the specified resource(s).
SAML assertion: **Conditions**

*(Optional)*: The assertion status may be dependent on:

- Additional information from a validation service.
- Other assertions being valid.

---

SAML assertion package

---

SAML assertion: **Advice**

*(Optional)*: Additional information that may be used to specify the assertions that were used to make a policy decision.

The Advice element is a **general container for any additional information** that does not affect the semantics or validity of the assertion itself.
SAML AssertionPackage

The `<SAMLAssertionPackage>` element is specified by the following schema:

```xml
<complexType name="SAMLAssertionPackageType">
  <!-- Basic information -->
  <attribute name="Version" type="string"/>
  <attribute name="AssertionID" type="uriReference"/>
  <attribute name="Issuer" type="string"/>
  <attribute name="IssueInstant" type="timeInstant"/>
  <attribute name="NotBefore" type="timeInstant"/>
  <attribute name="NotOnOrAfter" type="timeInstant"/>

  <element name="Claims" type="so:Claims" minOccurs="0"/>
  <element name="Conditions" type="so:Conditions" minOccurs="0"/>
  <element name="Advice" type="so:Advice" minOccurs="0"/>
</complexType>
```

SAML protocol

If a suitable assertion already exists, then that assertion may be returned in response to the request, without the responder having to create a new one.
XACML
(Extensible Access Control Markup Language)
What is XACML

- It is an XML specification for expressing **policies** for **information access** over the Internet
- XACML targets any object that can be **referenced** using XML
- XACML allows the assignment of privileges **directly** to users
- XACML does **not** specify the action primitive at all
- XACML specifications document has not been released yet

**XKMS**

**XML Key Management Specification**
**What is XKMS**

A protocol for:

- Distributing

- Registering public keys
**XKMS purpose**

<table>
<thead>
<tr>
<th>Without XKMS</th>
<th>With XKMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKI → Client</td>
<td>PKI</td>
</tr>
<tr>
<td>Client → PKI</td>
<td>Trusted Service</td>
</tr>
</tbody>
</table>

- Client complexity is high
- XKMS shields clients from PKI complexity

**XKMS Components**

- The XML Key Information Service Specification (X-KISS)
- The XML Key Registration Service Specification (X-KRSS)
### XKMS sub-protocol: X-KISS

- Allows a client to delegate part or all of the tasks required to process XML Signature to a Trust service.

- The underlying PKI may be based upon a different specification such as X.509/PKIX, SPKI or PGP.

### XKMS: Tiered Service Model

**KeyInfo**: Optional element that enables the recipient's to obtain the key needed to validate the signature.

- KeyInfo may contain:
  1. Keys
  2. Names
  3. Certificates

<table>
<thead>
<tr>
<th>Tiers</th>
<th><code>&lt;ds:keyinfo&gt;</code> processing</th>
<th><code>&lt;ds:keyinfo&gt;</code> Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tier 0</td>
<td>Done by Application</td>
<td>NA</td>
</tr>
<tr>
<td>Tier 1</td>
<td>Done by trust service</td>
<td>Done by Application</td>
</tr>
<tr>
<td>Tier 2</td>
<td>Done by trust service</td>
<td>Done by trust service</td>
</tr>
</tbody>
</table>
X-KISS: Tiered Service Model

Tier 0:

- XKMS is **not** deployed
- The client pulls PKI info from PKI server

![Diagram showing Tier 0: Get certificate flow from Client to Server]

X-KISS: Tiered Service Model

**Tier 1:** (Key Locating service)

- A client receives a signed XML document
- The client requests the trust server to obtain the public key parameters

![Diagram showing Tier 1: Key Name and Value flow among Client, Trust Service A, and Trust Service B]

Protected by XML Signature, SSL,IPSEC

Done only if info is not available in Server A
**X-KISS: Tiered Service Model**

**Tier 2: (Key Validating service)**

- A client receives a signed XML document
- The client queries the trust server to determine whether the signing key is trustworthy.
- The Trust Service builds a certificate trust path, then validates each certificate in the path against the relevant CRL

**XKMS sub-protocol: X-KRSS**

- X-KRSS permits management of information that is bound to a public key pair

  2 ways to generate a public key pair:
  1. In advance by the client, or
  2. On request by the service (to support key recovery)

- Services provided:
  1. Registration
  2. Revocation
  3. Key Recovery
XML Signature

Topics of the Presentation
- XKMS
- public key
- "credentials"
- User
- principal
- SAML
- "credentials"
- assertions
- XACML
- Resource
- Service Provider
- User
- principal
- assertions
- User
- principal
- "credentials"
- public key
- "credentials"
- User
- principal
- encrypted/signed XML doc
- XKMS
- XML signature
- XML encryption
Scope

- XML signature is composed of:
  1. Syntax used for representing the signature of Web resources (anything referenceble by a URI)
  2. Procedures for computing and verifying such signatures.

- XML Signatures are generated from a hash over a signature manifest (a collection of references to the objects being signed)

- XML signature does not address mechanisms for making statements or assertions.

XML signature

- The XML Signature data structures must be based on the RDF data model

- XML Signatures apply to any resource addressable by a locator including non-XML content

- XML Signatures may apply to a part or totality of an XML document

- XML Signatures are first class objects themselves and consequently must be able to be referenced and signed
XML digital signatures are represented by the **Signature element** which has the following structure:

```
<Signature>
  <SignedInfo>
    <CanonicalizationMethod/>
    <SignatureMethod/>
    (<Reference (URI=)? >
      <Transforms>?
        <DigestMethod>
          <DigestValue>
          </DigestValue>
        </DigestMethod>
      </Transforms>
    </Reference>)+
  </SignedInfo>
  <SignatureValue/>
  (KeyInfo)?
  (Object)*
</Signature>
```

**Signature element example**

```
<s01> <Signature Id="MyFirstSignature" 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#dsa-sha1"/>
    <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126"/>
    <Transforms>?
      <DigestMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    </Transforms>
    <DigestValue >j6lwx3rvEPO0vKtMup4NbeVu8nk=</ DigestValue >
  </Reference>
</SignedInfo>
  <SignatureValue >MC0CFFrVLrRlk=...</ SignatureValue >
  <KeyInfo>
    <KeyValue>
      <DSAKeyValue >...</DSAKeyValue >
    </KeyValue>
  </KeyInfo>
</Signature>
```

The key to be used to validate the signature info that is actually signed
**Signature Process**

1. Core Generation
   - Reference Generation
   - Signature Generation

2. Core Validation
   - Reference Validation
   - Signature Validation

---

**Signature Process: Core Generation**

1. Core Generation
   a) Reference Generation: Creating a Reference element for each data object to be signed
   b) Signature Generation

![Diagram of Signature Process: Core Generation]

- Data object \(\rightarrow\) Transform \(\rightarrow\) Transformed Data object \(\rightarrow\) Calculate digest value \(\rightarrow\) Data object digest
- Include: Id of the data object, transform elements, digest algorithm, DigestValue
- Create a Reference element
- Reference element
1. Core Generation (continues)
   a) Reference Generation
   b) Signature Generation

   **Signature Process: Core Generation**
   
   Reference element
   Specify: SignatureMethod,
   Specify: CanonicalizationMethod
   SignedInfo element
   Canonicalize and then
   Calculate SignatureValue
   Signature Value
   Create
   Signature element
   SignedInfo{References},
   KeyInfo
   SignatureValue
   Signature element

2. Core Validation
   a) Reference Validation: The verification of the digest contained in each Reference
      in SignedInfo
   b) Signature Validation

   **Signature Process: Core Validation**
   
   SignedInfo{References}
   KeyInfo
   SignatureValue
   Reference element
   CanonicalizationMethod
   transform
   Reference element
   Obtain data object to be digested
   Data object
calculate digest value
   Data Object
   Compare digest value
   Data object
   Digest
   Reference validation result
   Signature element
Topics of the Presentation

2. Core Validation
   a) Reference Validation
   b) Signature Validation: Cryptographic signature validation of the signature calculated over SignedInfo

XML security protocols and Framework for Security Engineering
References

3. A Technical Introduction to XML by Norman Walsh (http://nwalsh.com/docs/articles/xml)
5. SAML Specification Version 00 draft-sstc-saml-spec-00.doc (No date was specified): (http://www.oasis-open.org/committees/security/docs/draft-sstc-saml-spec-00.PDF)
6. XACML: (http://www.oasis-open.org/committees/xacml/docs/)
7. XML Key Management Specification (XKMS), 30 March 2001: (http://www.w3.org/TR/xkms/)
9. XML-Signature Syntax and Processing: (http://www.w3.org/TR/2001/PR-xmldsig-core-20010820)