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

XML in brief

- XML is a 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.
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>
```

SAML
(Secure Assertion Markup Language)

Main XML security protocols

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)

SAML: Scenario #2: Authorization Service
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

**Claims**
- “DecisionClaim”: Access permissions specified in the request identified by the corresponding RequestID were either permitted, denied or could not be determined
- “AuthenticationClaim”: Specified subject has been authenticated
- “AttributeClaim” element: Specified subject has the specified attribute(s) specified by a URI
- “AuthorizationClaim”: Specified subject is authorized to perform the specified operation(s) on the specified resource(s).

**Conditions**
(Optional): The assertion status may be dependent on:
- Additional information from a validation service.
- Other assertions being valid.

**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.

The `<SAMLAssertionPackage>` element is specified by the following schema:
**XACML**
(Extensible Access Control Markup Language)

**Topics of the Presentation**
- XKMS
- public key “credentials”
- User i/principal
- SAML
- “credentials” assertions
- User j/principal
- XML signature
- XML encryption

**XKMS**
XML Key Management Specification

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

What is XKMS
A protocol for:
- Distributing
- Registering public keys
XKMS purpose

Without XKMS

- Directory
- Client
- PKI

Client complexity is high

With XKMS

- PKI
- XKMS
- Trust
- Service
- Client
- Client

XKMS shields clients from PKI complexity

XKMS Components

- The XML Key Information Service Specification (X-KISS)
- The XML Key Registration Service Specification (X-KRSS).

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

Tiers | cd:ds:KeyInfo: processing | cd:ds:KeyInfo: Validation
--- | --- | ---
Tier 0 | Done by Application | NA
Tier 1 | Done by trust service | Done by Application
Tier 2 | Done by trust service | Done by trust service

XKMS: Tiered Service Model

Tier 0:

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

Tier 1: (Key Locating service)

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

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.
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**

- XML signature is composed of:
  1. Syntax used for representing the signature of Web resources (anything referenceable 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:

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

**Signature element example**

```xml
<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>
        <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
  <KeyInfo>
    <KeyValue>
      <DSAKeyValue>
        <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
      </DSAKeyValue>
    </KeyValue>
  </KeyInfo>
</Signature>
```

**Signature Process: Core Generation**

1. **Core Generation**
   a) **Reference Generation**: Creating a Reference element for each data object to be signed
   b) **Signature Generation**: Specify: SignatureMethod, CanonicalizationMethod

```text
Reference element
  Specify: SignatureMethod, CanonicalizationMethod
  Calculate digest value
  Include: ID of the data object, transform elements, digest algorithm, DigestValue
  Create a Reference element
```

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

```text
Reference validation result
  Compare digest value
  Calculate digest value
  Data Object
```
**Topics of the Presentation**

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

![Diagram](attachment://diagram.png)

**References**

4. SAML Specifications June 20, 2001: draft-sstc-ftf3-saml-spec-00.txt (A very poorly written unstable document): [http://xml.coverpages.org/draft-sstc-ftf3-saml-spec-00.txt](http://xml.coverpages.org/draft-sstc-ftf3-saml-spec-00.txt)
5. SAML Specification Version 00 draft-sstc-ftf3-saml-spec-00.doc (No date was specified): [http://www.oasis-open.org/committees/security/docs/draft-sstc-ftf3-saml-spec-00.PDF](http://www.oasis-open.org/committees/security/docs/draft-sstc-ftf3-saml-spec-00.PDF)
9. XML Signature Syntax and Processing: [http://www.w3.org/2001/04/xmlsig-core](http://www.w3.org/2001/04/xmlsig-core)