Java Security

Web Services Security
(Overview)

Lecture 9
Java 2 Cryptography

- Java provides API + SPI for crypto functions
  - Java Cryptography Architecture
    - Security related core classes
      - Access control and cryptography
  - Java Cryptography Extension
    - Other core classes
      - Message digest, digital signatures, certificate management
      - Key exchange, MAC
JCA + JCE

- **Engine**
  - Abstract cryptographic service: E.g., message digest, digital signatures
  - To provide cryptographic operations
  - To generate or supply the crypto material
  - To generate and manage data objects (certificates or keys – keystores)
    - Use instances of engine class for crypto operations

- **Algorithm**
  - Implementation of an engine: Eg. MD5 for MessageDigest

- **Provider**
  - (set of) packages that supply concrete implementation of a subset of the cryptographic services (DS, MD, etc.)
JCA + JCE Principles

- Provider based architecture
- Vendors can register implementations of algorithms
- Providers can be configured declaratively so the application code does not need to change
- Allows different implementations to be found at runtime

- Engine Class
- SPI class
  - Implementations expose the same API

Java Program Requesting MD5 Message Digest

<table>
<thead>
<tr>
<th>MD5 Implementation by Provider A</th>
<th>MD5 Implementation by Provider B</th>
</tr>
</thead>
</table>

Implementation independence

Algorithm Independence
JCA + JCE Principles

- New algorithms can be easily plugged in
  - Has to be compliant with the MessageDigest API

- Various implementation can
  - work with one another
  - Use one another’s keys
  - Verify one another’s messages

![Diagram of Algorithm Extensibility and Implementation Interoperability]
Providers

● SPI is
  ● Key to pluggability, extensibility and module independence
  ● It is a set of Java-language interfaces and abstract classes for cryptographic services

● A Provider is a pluggable modules
  ● Provides concrete implementations of some SPI methods
    ● `java.security` and `javax.crypto` and their subpackages contain many SPI interfaces that JCA and JCE providers can implement
Providers
Engine and SPI

- Engine classes are the interfaces between the user code and the implementations.
- Implementations are found at runtime.
Engine and SPI

- The engine class calls the SPI class methods
  - SPI class method names begin with “engine”
  - Implementation of abstract SPI done by providers
Enterprise Security for Web Services

- **XML**
  - Simplicity and flexibility
  - Facilitates B2B messaging
  - Security is a big concern
    - Structured semantics and schema-driven nature

- **XML security technologies are available**
  - Encryption
    - Elements, sections
  - Digital signatures
    - All or parts – by one or more entities
  - Access control
Web Service

- Web service
  - Is a an interface that describes a collection of network-accessible operations based on open internet standards
  - Potential to enable application integration at a higher level of the protocol stack
    - based on Web Services standards
  - XML
  - Simple Object Access Protocol
  - Web Services Description Language (WSDL)
  - Universal Description, Discovery and Integration
SOAP

- Simple, lightweight and extensible XML-based mechanism for exchanging structured data between network applications
- Consists of:
  - An envelop
    - What is in the message and who should deal with it
  - A set of coding rules
    - Serialization mechanism that be used to exchange instance of application defined data types
SOAP

- It supports modular architecture
  - Allows defining the following in separate documents
    - WS Addressing Specification (WS-Addressing)
    - WS Security Specification (WS-Security)

- A SOAP envelope is defined in
  - **Envelope** XML element
  - Consists of two parts:
    - Header: adds features to the messages
      - Meta information can be added to the message
      - E.g., transaction IDs, message routing information, message security
    - Body: mechanism for exchanging information
Security Technologies

- **XML Signature**
  - Validation of the messages and non-repudiation
- **SAML**
  - AuthM + Security Srvices ML
    - Authentication + Authorization profile information
  - Common language for sharing of security services between companies for B2B/B2C transactions
- **XML Encryption**
  - Encrypting of XML fragments
- **WS-Security**
  - Set of SOAP extensions that can be used when building WS to implement integrity and confidentiality
XML Signature

- IETF and W3C standard for digitally signing all or some part of the XML document

- XML Signature
  - Is itself a piece of XML – defined by a schema
  - Contain references – URIs – to what is being signed
    - URIs – within the document or external to it
  - A singled XML document may have multiple signatures
XML Signature Structure

- XML Signature contains four major items:
  - A set of pointers (references) to things to be signed
  - The actual signature
  - (Optional) The key (or a way to look up the key) for verifying the signature
  - (Optional) An Object tag that can contain miscellaneous items not included in the first three items

```xml
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <Reference URI="http://www.foo.com/secureDocument.html" />
  </SignedInfo>
  <SignatureValue>...</SignatureValue>
  <KeyInfo>...</KeyInfo>
</Signature>
```
XML Signature: Enveloping Signature

<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo> <Reference URI="#111" /> </SignedInfo>
  <SignatureValue>...</SignatureValue>
  <KeyInfo>...</KeyInfo>
  <Object>
    <SignedItem id="111">Stuff to be signed</SignedItem>
  </Object>
</Signature>
XML Signature: Enveloped Signature

```xml
<PurchaseOrder id="po1">
  <SKU>125356</SKU>
  <Quantity>17</Quantity>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo> <Reference URI="#po1" /> </SignedInfo>
    <SignatureValue>...</SignatureValue>
    <KeyInfo>...</KeyInfo>
  </Signature>
</PurchaseOrder>
```
XML Signature: Detached Signature

Detached Signature within same XML Document

```xml
<PurchaseOrderDocument>
  <PurchaseOrder id="po1">
    <SKU>12366</SKU>
    <Quantity>17</Quantity>
  </PurchaseOrder>
</PurchaseOrderDocument>
```

```xml
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <Reference URI="#po1"/>
  </SignedInfo>
  <SignatureValue>...</SignatureValue>
  <KeyInfo>...</KeyInfo>
</Signature>
```

```xml
</PurchaseOrderDocument>
```
**XML Signature: Detached Signature**

Can also reference external source

```xml
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <Reference URI="http://www.foo.com/picture.jpg"/>
  </SignedInfo>
  <SignatureValue>...</SignatureValue>
  <KeyInfo>...</KeyInfo>
</Signature>
```
XML Encryption Structure

- Could encompass an entire document of other XML structure (similar to Enveloping structure)
- Could contain a pointer to a detached resource
XML Encryption: Example

<Employee>
  <Name>Dave Remy</Name>
  <SocialSecurityNumber>
    <EncryptedData Type="http://www.w3.org/2000/09/xmldsig#content">
      <EncryptionMethod Algorithm="...">
        <CipherData><CipherValue>. . .</CipherValue>
      </CipherData>
    </EncryptedData>
  </SocialSecurityNumber>
  <Salary>
    <EncryptedData Type="http://www.w3.org/2000/09/xmldsig#content">
      <EncryptionMethod Algorithm="...">
        <CipherData><CipherValue>. . .</CipherValue>
      </CipherData>
    </EncryptedData>
  </Salary>
</Employee>
<Employee>
  <Name>Dave Remy</Name>
  <SocialSecurityNumber>
    <EncryptedData id="socsecnum" Type="http://www.w3.org/2000/09/xmldsig#content">
      <EncryptionMethod Algorithm=".
      ." />
      <CipherData> <CipherValue>. .</CipherValue> </CipherData>
    </EncryptedData>
  </SocialSecurityNumber>
  <Salary>
    <EncryptedData id="salary" Type="http://www.w3.org/2000/09/xmldsig#content">
      <EncryptionMethod Algorithm=".
      ."> </CipherData> <CipherValue>. .</CipherValue> </CipherData>
    </EncryptedData>
  </Salary>
  <EncryptedKey>
    <EncryptionMethod Algorithm=".
    ." />
    <CipherData> <CipherValue>. .</CipherValue> </CipherData>
    <ReferenceList>
      <DataReference URI="#socsecnum" />
      <DataReference URI="#salary" /> </ReferenceList>
  </EncryptedKey>
</Employee>
SAML

- Enables portable identities and the assertions that these identities want to make
  - Assertion: authentication; authorization

- SAML is important for WS
  - is a standard XML format – all normal XML tools apply to SAML
  - Includes a standard message exchange protocol
  - Specifies the rules for how it is transported – making interoperability explicit at the specification level
  - Expression of security in the form of assertions about subjects (different from Certification authority based approach) – facilitated Single-Sign ON
SAML - scenario

- Federated identity
SAML

- Defines three types of assertions
  - **Authentication**
    - States that a particular auth. authority has authenticated the subject
      - Using a particular process
      - At a particular time (+ validity)
  - **Authorization**
    - States that a particular authority has granted/denied permissions on particular resource (+time)
  - **Attributes**
    - Provides qualifying information about either an authentication or authorization assertion
SAML – how it works

- Three XML based mechanisms and their relationship

  - XML schema + definition
  - Rules on using assertions
SAML Example

<saml:Assertion>
  MajorVersion="1" MinorVersion="0"
  AssertionID="192.168.0.1.12345"
  Issuer="Company.com"
  IssueInstant="2004-01-21T10:02:00Z">
  <saml:Conditions>
    NotBefore="2004-01-21T10:02:00Z"
    NotAfter="2004-01-21T10:09:00Z" />
  <saml:AuthenticationStatement>
    AuthenticationMethod="password"
    AuthenticationInstant="2004-01-21T10:02:00Z"
  </saml:AuthenticationStatement>
  <saml:Subject>
    <saml:NameIdentifier
      SecurityDomain="Company.com"
      Name="jothy" />
  </saml:Subject>
</saml:Assertion>
SAML protocol

- SAML assertions are sent to the authentication and authorization authorities
SAML Authorization/Attribute Assertions

<saml:Assertion ...>
  <saml:AttributeStatement>
    <saml:Subject>...</saml:Subject>
    <saml:Attribute>
      AttributeName="PaidStatus"
      AttributeNamespace="http://smithco.com">
    <saml:AttributeValue>
      PaidUp
    </saml:AttributeValue>
  </saml:Attribute>
  <saml:Attribute>
    AttributeName="CreditLimit"
    AttributeNamespace="http://smithco.com">
    <saml:AttributeValue xsi:type="my:type">
      <my:amount currency="USD">500.00
    </my:amount>
    </saml:AttributeValue>
  </saml:Attribute>
</saml:AttributeStatement>
</saml:Assertion>

<saml:Assertion ...>
  <saml:AuthorizationStatement
    Decision="Permit"
    Resource="http://jonesco.com/doit.cgi">
    <saml:Subject>...</saml:Subject>
      <saml:Action>
      </saml:Action>
    </saml:AuthorizationStatement>
  </saml:Assertion>
SAML Architecture
SAML Binding

- Requires SOAP over HTTP as one binding
- SOAP Binding
  - SAML information is contained inside the SOAP
- SAML Profile
  - Describes how SAML assertions are embedded into and extracted from a framework/protocol
    - Browser profile of SAML
    - SAML profile SOAP
    - WS-Security
WS-Security

- Focuses on applying existing security technologies to SOAP message
  - X.509 certificates
  - SAML assertions
  - XML Signatures
  - XML Encryption

- GOAL: Secure the SOAP
  - No matter where it goes
  - No matter how long it lives
HTTP Transport Security Versus Message Security

- HTTP Transport Security
  - Authentication at the time secure pipe is created
  - Confidentiality/Integrity in the pipe only
HTTP-TS
Pros and Cons

Pros

- **Mature**: Tried and true
- **Support**: Supported by most servers and clients
- **Understood**: Understood by most system administrators
- **Simpler**: Generally simpler than message-level security alternatives

Cons

- **Point to Point**: Messages are in the clear after reaching SSL endpoint
- **Waypoint visibility**: Cannot have partial visibility into the message
- **Granularity**: Cannot have different security for messages in and messages out
- **Transport dependent**: Applies only to HTTP
Message Security Pros and Cons

Pros

- **Persistent**: Allows the message to be self-protecting
- **Selective**: Portions of the message can be secured to different parties
- **Flexible**: Different security policy can be applied to request and response - Transport independent

Cons

- **Immature**: standard, tools
- **Complex**: encompasses many other standards including XML Encryption, XML Signature, X.509 certificates, and many more
Web Services Security Stack