

Personenkontrollierte Auslösung MOA-SS

Signatur im E-Government

Version 1.0.0, 08.11.2006

DI Arne Tauber – arne.tauber@egiz.gv.at

Zusammenfassung:

In diesem Dokument wird eine Erweiterung von MOA-SS beschrieben, die es ermöglicht Signaturen auf Basis von verschiedenen Identifikationsmethoden (z.B. über Bürgerkarte / Benutzername, Passwort, etc.) auszulösen. Das im Umfang dieser Dokumentation mitgelieferte Demo zeigt eine Auslösung einer Signatur auf Basis einer eindeutigen Identifikation des Signators mittels MOA-ID.

Die Erweiterung von MOA-SS wurde auf einer Abstraktionsebene realisiert, die auch eine session-basierte oder mengenbasierte Auslösung von Signaturen ermöglicht.

Inhaltsverzeichnis:

1.1	Grundlagen	5
1.2	Funktionsbeschreibung	5
1.3	Voraussetzungen zur Nutzung der Demoanwendung	7
3.1	Systemanforderungen	11
3.2	Dateien	11
3.3	Installation	11
3.4	Konfiguration	11
	Anhang A	15

Anmerkung: Zur besseren Lesbarkeit wurde in diesem Dokument teilweise auf geschlechtsspezifische Formulierungen verzichtet. Die verwendeten Formulierungen richten sich jedoch ausdrücklich an beide Geschlechter.

Personenbezogene Auslösung MOA-SS Signatur
Signatur im E-Government

Abbildungen

Abbildung 1: <i>Authenticator</i> Element	5
Abbildung 2: <i>AuthConstraint</i> Element	6
Abbildung 3: Ablauf der personenbezogenen Signaturauslösung über MOA-SS und MOA-ID	7
Abbildung 4: Webinterface für die personenbezogene Auslösung	8
Abbildung 5: Automatisch befülltes Feld nach erfolgreiche MOA-ID Anmeldung.....	9
Abbildung 6: Erfolgreiches Auslösen einer MOA-SS Signatur.....	9
Abbildung 7: Fehlgeschlagene Authentifizierung mit falschem Code.....	10

Revision History

Version	Datum	Autor(en)	
1.0.0	08.11.2006	Arne Tauber (EGIZ)	Initialversion

1 Kurzbeschreibung

1.1 Grundlagen

Das Serversignaturmodul MOA-SS kann für die hardware- oder softwarebasierte Erstellung von Signaturen verwendet werden. Das Modul ist für den Einsatz im Backoffice Bereich im Kontext von Fachapplikationen der öffentlichen Verwaltung konzipiert. In der aktuellen Fassung des Moduls ist daher nur eine Authentifizierung seitens der Fachapplikation vorgesehen.

In vielen Fällen macht es jedoch Sinn, eine serverbasierte Signatur auf Basis einer eindeutigen Identifikation des Signators auszulösen (personenbezogene und nicht applikationsbezogene Identifikation). Als Beispiel sei hier der elektronische Akt (ELAK) angeführt.

In diesem Dokument wird eine Erweiterung von MOA-SS beschrieben, die es ermöglicht Signaturen auf Basis von verschiedenen Identifikationsmethoden (z.B. über Bürgerkarte / Benutzername, Passwort, etc.) auszulösen. Das im Umfang dieser Dokumentation mitgelieferte Demo zeigt eine Auslösung einer Signatur auf Basis einer eindeutigen Identifikation des Signators mittels MOA-ID.

Die Erweiterung von MOA-SS wurde auf einer Abstraktionsebene realisiert, die auch eine session-basierte oder mengenbasierte Auslösung von Signaturen ermöglicht.

1.2 Funktionsbeschreibung

Bis dato bot MOA-SS nur eine Authentifizierung auf Basis von SSL Clientzertifikaten an. Das Konzept zur Erweiterung von MOA-SS sieht vor, dass Signaturanfragen an MOA-SS mit eigenen request-basierten Modulen zur Authentifizierung/Identifizierung gegengeprüft werden können. Dies schafft die Möglichkeit, alle Komponenten und Aspekte eines HTTP Requests für eine proprietäre Authentifizierungslösung heranzuziehen. Solche Komponenten könnten z.B. sein:

- SSL Clientzertifikat
- HTTP Header
- Verbindungsparameter (IP-Adresse, etc.)

1.2.1 Schemaerweiterung MOA-SS Konfiguration

Das Schema der Konfigurationsdatei von MOA-SS wurde um einige optionale Elemente erweitert, die es ermöglichen, proprietäre Authentifizierungsmodule zu integrieren.

Das *SignatureCreation* Element wurde um das optionale *Authenticator* Element erweitert, mit dem oben erwähnte Module registriert werden können.

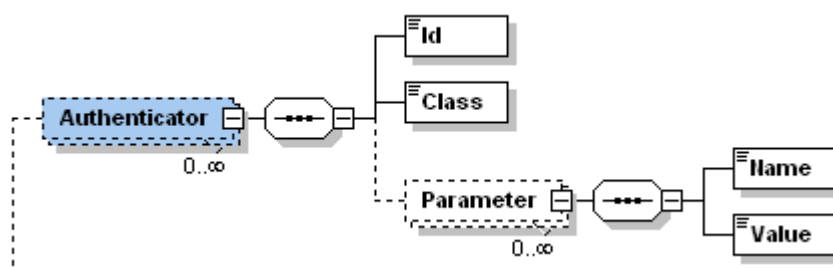


Abbildung 1: *Authenticator* Element

Für jedes Authentifizierungsmodul muss eine eindeutige ID (Element *Id*) angegeben werden. Über diese ID wird das Modul selektiert, falls eine bestimmte Schlüsselgruppe mit dem Modul geschützt werden soll. Im *Class* Element muss der qualifizierte Klassenname des Authentifizierungsmoduls eingetragen werden. Diese Klasse muss das Interface *at.gv.egovernment.moa.spss.server.invoke.RequestAuthenticator* (siehe Anhang A) implementieren. Optional können weitere Parameter über das optionale Element *Parameter* dem Modul bei dessen Initialisierung übergeben werden. Ein solcher Parameter besteht aus einem Namen-Wert Paar. Beide Angaben sind obligatorisch.

Für jedes *KeyGroupMapping* Element kann zusätzlich eine Menge von *AuthConstraint* Elementen definiert werden, die die Art und Weise der Authentifizierung definieren. So ist es auch möglich für eine bestimmte Menge von Schlüsseln mehrere Authentifizierungsmethoden zu definieren (z.B. MOA-ID und/oder Benutzername/Passwort).

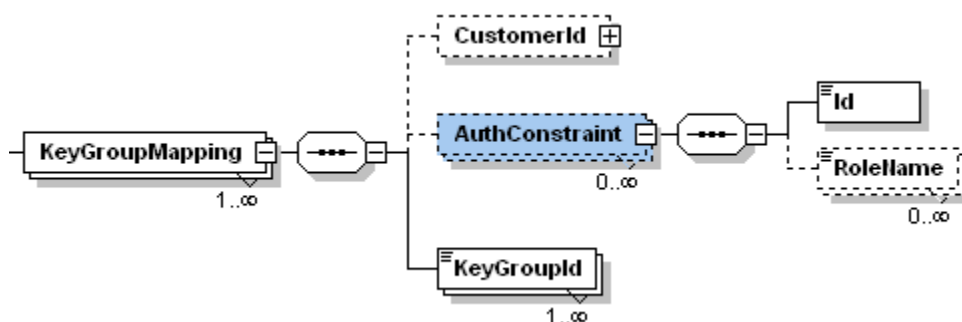


Abbildung 2: *AuthConstraint* Element

Für jedes *AuthConstraint* Element muss die *Id* eines *Authenticator* Elements (wie oben beschrieben) angegeben werden. Dieser *Authenticator* wird dann für die Authentifizierung des Requests herangezogen. Zusätzlich können optional eine Menge an Rollen an den *Authenticator* übergeben werden. Dies macht jedoch nur Sinn, falls der *Authenticator* eine rollenbasierte Authentifizierung unterstützt.

1.2.2 Beispiel: MOA-SS Signatur mittels Identifikation über MOA-ID

Es wurde ein Demonstrator entwickelt, der eine personenbezogene Signatur über MOA-SS auslöst. Basierend auf obiger MOA-SS Erweiterung wurde ein MOA-ID *Authenticator* entwickelt, der eine Abfrage an MOA-ID bezüglich Identifikationsdaten für ein gegebenes *SAMLArtifact* durchführt. Ein derartiges Artfakt wird nach einer erfolgreichen Anmeldung an MOA-ID mit der Bürgerkarte erzeugt. Mit Hilfe dieses Artfakts kann eine Applikation eine einmalige Abfrage der Identifikationsdaten an MOA-ID durchführen. Im Falle einer Single-Sign-On Lösung von MOA-ID wäre die Abfrage mittels Artfakt für eine bestimmte Zeitspanne (Session) durchführbar.

Der Demonstrator ist so realisiert, dass nach einer erfolgreichen MOA-ID Authentifizierung des Signators das von MOA-ID retournierte SAML Artfakt als HTTP Header (Header Name *SAMLArtifact*) im Request an MOA-SS mitgeschickt wird. Der MOA-ID *Authenticator* liest diesen Header aus und führt eine Abfrage der Identifikationsdaten an MOA-ID durch. Falls Identifikationsdaten vorliegen wird die Signatur ausgelöst, andernfalls nicht. Der *Authenticator* lässt somit jede erfolgreiche Identifikation zu einer Signaturauslösung durch. Ein Decision Point für eine Filterung und Selektierung von Identifikationsdaten könnte noch optional eingebaut werden.

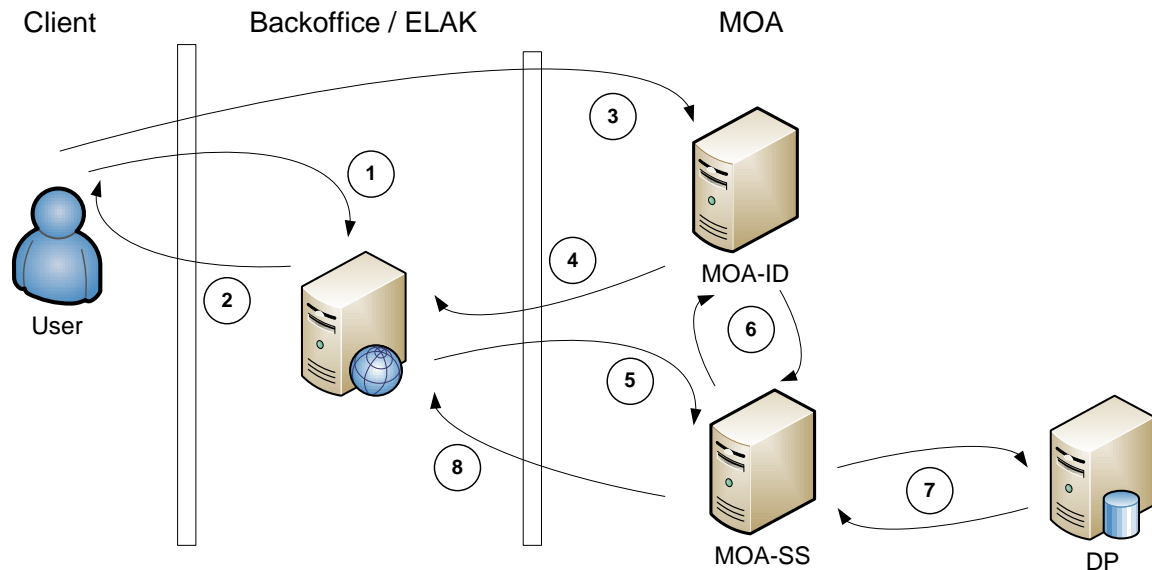


Abbildung 3: Ablauf der personenbezogenen Signaturauslösung über MOA-SS und MOA-ID

1. Der Benutzer (Signator) startet in seinem Client (Webbrowser) die Auslösung der Signatur
2. Die Backoffice Applikation erkennt, dass der Benutzer noch nicht authentifiziert wurde und
3. leitet diesen zu einer Identifikation an MOA-ID um.
4. Nach erfolgreicher Identifikation wird das von MOA-ID erzeugte SAML Artifakt an die Backoffice Applikation weitergereicht.
5. Diese sendet nun eine Signaturrequest an MOA-SS, der einen HTTP Header mit dem Artifkat enthält.
6. Der in MOA-SS registrierte MOA.ID Authenticator extrahiert den Wert des HTTP Headers und führt mit diesem eine Abfrage der Identifikationsdaten an MOA-ID durch.
7. Optional könnte (was jedoch der Demonstrator nicht implementiert) an einem Decision Point eine Abfrage gestartet werden, ob der authentifizierte Benutzer mit seinen Identifikationsdaten überhaupt eine Signatur auslösen darf.
8. Falls die Überprüfung der Identifikationsdaten erfolgreich verlaufen ist, wird die Signatur erstellt und an die Backoffice Applikation retourniert.

1.3 Voraussetzungen zur Nutzung der Demoanwendung

- Ein Webbrowser
- Java 1.4
- Ein Application Server (empfohlen wird Tomcat 5.0)
- MOA Komponenten: MOA-SS (inkl. Authenticator Upgrade), MOA-ID

2 Anwendungsbeschreibung

Der nachfolgenden Anwendungsbeschreibung wird folgende Terminologie bzw. werden folgende Parameter zu Grunde gelegt:

Parameter

Name	Wertebereich	Beschreibung
%CONTAINER%	n/a	Bezeichnet den verwendeten Servlet Container.
%CONTEXT%	n/a	Bezeichnet den Servlet Context, der dem Container zugeordnet wird.




Die Demoanwendung wird durch einen Aufruf folgender Adresse in einem Webbrowser gestartet:

%CONTAINER%/%CONTEXT%

z.B. <https://demo.egiz.gv.at/auth-sign/>

Es erscheint anschliessend das Webinterface für die personenbezogene Auslösung der MOA-SS Signatur.

Personenbezogene Auslösung MOA-SS Signatur


Info Bitte beachten Sie: * Feld muss ausgefüllt sein  Information und Hilfe zum Ausfüllen  Interaktive Ausfüllhilfe
! Hinweis auf Fehler Zutreffendes ankreuzen oder  auswählen

Personenbezogene Auslösung einer Serversignatur über MOA-SS

Identifikationscode:

Achtung: dieser Code ist nur EINMAL für die Auslösung einer Signatur gültig. Wenn Sie weitere Signaturen auslösen möchten, müssen Sie anmelden.

Um die Signatur über MOA-SS auslösen zu können, müssen Sie sich zuerst über MOA-ID anmelden um den entsprechenden Anmeldecode zu Formen der Bürgerkarte gibt, müssen Sie entscheiden, welche Sie bei der Anmeldung verwenden wollen.

 Bürgerkarten Software ist bereit!




Abbildung 4: Webinterface für die personenbezogene Auslösung

Das Interface stellt ein Textfeld für die Eingabe des einmaligen Identifikationscodes (SAML Artifikat) zur Verfügung, das für die Auslösung der Signatur benötigt wird. Dieser Code ist nur einmalig gültig und somit muss für jede weitere Auslösung einer Signatur eine erneute Anmeldung an MOA-ID durchgeführt werden. Das Textfeld muss jedoch nicht manuell befüllt werden, sondern wird nach der Anmeldung an MOA-ID automatisch mit dem Code (Artifikat) befüllt.

Die Anmeldung an MOA-ID kann entweder mit der Chipkarte (E-Card, Bankomatkarte, ...) oder mittels A1-Signatur durchgeführt werden.

Nach erfolgreicher Anmeldung wird das Feld wie in folgender Abbildung automatisch befüllt.

Personenbezogene Auslösung MOA-SS Signatur

Info
Bitte beachten Sie:

- * Feld muss ausgefüllt sein
- ! Hinweis auf Fehler
- i Information und Hilfe zum Ausfüllen
- Zutreffendes ankreuzen oder auswählen
- 🔗 Interaktive Ausfüllhilfe

Identifikationscode:

Achtung: dieser Code ist nur EINMAL für die Auslösung einer Signatur gültig. Wenn Sie weitere Signaturen auslösen möchten, müssen S
 anmelden.

Um die Signatur über MOA-SS auslösen zu können, müssen Sie sich zuerst über MOA-ID anmelden um den entsprechenden Anmeldecode z
 Formen der Bürgerkarte gibt, müssen Sie entscheiden, welche Sie bei der Anmeldung verwenden wollen.



Bürgerkarten
Software
ist bereit!



Abbildung 5: Automatisch befülltes Feld nach erfolgreiche MOA-ID Anmeldung

Die Signatur kann nun durch Drücken auf den Button „Signatur auslösen“ erstellt werden. Es öffnet sich ein neues Fenster und die erstellte XML Signatur wird angezeigt.

```

- <soapenv:Envelope>
- <soapenv:Body>
- <CreateXMLSignatureResponse>
- <SignatureEnvironment>
- <dsig:Signature Id="signature-1-1">
- <dsig:SignedInfo>
  <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
  <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
  - <dsig:Reference Id="reference-1-1" URI="#xpointer(id('signed-data-1-1-1')/node())">
    <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <dsig:DigestValue>eLODyeiWFbAkQKwhrR23jtcgu4k=</dsig:DigestValue>
  </dsig:Reference>
</dsig:SignedInfo>
- <dsig:SignatureValue>
  cqfgfUztUdqCzqfU/sr9b8SMrMVAcuwsOAgspk9znJTbnLuw/Umc+HCgfQ/nVsa/
  gZ+prKHozhQDpkyINzRMqnyCW6PofJwFZyXfbPSWzFP66DnpdIPZlvuXLUABQ5P5 CRJOiU66C5E
</dsig:SignatureValue>
- <dsig:KeyInfo>
- <dsig:X509Data>
  - <dsig:X509Certificate>
    MII EW D C C A 0 C g A w I B A g I U E x Q I G B E Y i Y o p / q B R X S S E A W f g b N s w D Q Y J K o Z I h v c N A Q E F
    D O A 1 E C A D M I D A M P A L E T I B C W I A V N L O C O Z E I I B E F I L O C I Z I E R M E T I B O V
  
```

Abbildung 6: Erfolgreiches Auslösen einer MOA-SS Signatur

Falls ein falscher Code eingegeben wurde, erscheint die Fehlermeldung, dass der Signaturschlüssel nicht zur Verfügung steht. Ein Schlüssel steht erst dann für die Auslösung einer Signatur zur Verfügung, falls die Authentifizierung erfolgreich war.

```
- <soapenv:Envelope>  
  - <soapenv:Body>  
    - <CreateXMLSignatureResponse>  
      - <ErrorResponse>  
        <ErrorCode>2231</ErrorCode>  
        <Info>Die Schlüsselgruppe ist nicht verfügbar</Info>  
      </ErrorResponse>  
    </CreateXMLSignatureResponse>  
  </soapenv:Body>  
</soapenv:Envelope>
```

Abbildung 7: Fehlgeschlagene Authentifizierung mit falschem Code

3 Deployment

3.1 Systemanforderungen

Es gelten folgende Anforderungen an die Installationsplattform bzw. an die Client-komponenten (die angegebenen Versionen entsprechen der getesteten Umgebung):

3.1.1 Server-Komponenten

- JDK 1.4.2
- Application Server (z.B. Tomcat 5.0)
- MOA-SS Server für Signaturerstellung mit Upgrade für die Integration von proprietären Authentifizierungsmodulen
- MOA-ID

3.1.2 Client-Komponenten

- Browser (z.B. Internet Explorer oder Mozilla Firefox)

3.2 Dateien

Name	Beschreibung
auth-sign.jar	Diese Datei enthält den MOA-ID Authenticator, der in MOA-SS registriert wird, sowie alle benötigten Hilfsklassen.
auth-sign.war	Dieses Webarchiv enthält die Demonstrationsapplikation (Beispiel für eine Backofficeapplikation)

3.3 Installation

Die Installation von MOA-SS und MOA-ID wird hier nicht explizit erläutert. Diese kann in (Österreich, MOA: Serversignatur (SS), 2006) und (Österreich, MOA: Identifikation (ID), 2006) nachgelesen werden.

Nachdem MOA-SS und MOA-ID erfolgreich installiert wurden, muss die Datei *auth-sign.jar* in das WEB-INF/lib Verzeichnis des MOA-SS Webkontexts installiert werden. Damit kann das Authentifizierungsmodul von MOA-SS dynamisch geladen werden.

Die Demoapplikation wird deployed, indem die Datei *auth-sign.war* in das webapps Verzeichnis des Tomcat Application Servers gegeben wird. Das Webarchiv wird anschliessend automatisch entpackt und die Demoapplikation geladen.

3.4 Konfiguration

3.4.1 MOA-ID Konfiguration

Die Demoapplikation muss bei MOA-ID registriert werden, damit das Artefakt an diese übergeben werden kann. Der Eintrag in der Konfigurationsdatei von MOA-ID lautet:

```
<OnlineApplication publicURLPrefix="%CONTAINER%/%CONTEXT%/">  
  <AuthComponent/>  
</OnlineApplication>
```

Beispielsweise:

```
<OnlineApplication publicURLPrefix="https://demo.egiz.gv.at/auth-  
sign/">  
  <AuthComponent/>  
</OnlineApplication>
```

3.4.2 MOA-SS Konfiguration

Der Eintrag für die Registrierung des MOA-ID Authenticators in der MOA-SS Konfigurationsdatei lautet:

```
<cfg:Authenticator>  
  <cfg:Id>MOAID</cfg:Id>  
  <cfg:Class>at.gv.egiz.authsign.MOAIDAuthenticator</cfg:Class>  
  <cfg:Parameter>  
    <cfg:Name>ServiceURL</cfg:Name>  
    <cfg:Value>http://localhost:8080/moa-id-  
auth/services/GetAuthenticationData</cfg:Value>  
  </cfg:Parameter>  
</cfg:Authenticator>
```

Achtung: Der Parameter ServiceURL muss noch der URL des MOA-ID Servers angepasst werden. Falls MOA-ID auf einem anderen Serverport konfiguriert wurde oder unter einem anderen Kontext läuft, muss der Parameter angepasst werden.

Falls ein Signaturschlüssel konfiguriert wurde (hier wird angenommen der Name der Keygroup sei *demo*), kann ein Keygroupmapping für diesen Schlüssel mit dem oben konfigurierten Authenticator eingerichtet werden.

```
<cfg:KeyGroupMapping>  
  <cfg:AuthConstraint>  
    <cfg:Id>MOAID</cfg:Id>  
  </cfg:AuthConstraint>  
  <cfg:KeyGroupId>demo</cfg:KeyGroupId>  
</cfg:KeyGroupMapping>
```

Damit ist der Schlüssel *demo* durch eine Authentifizierung über MOA-ID abgesichert.

3.4.3 Konfiguration der Demoapplikation

Die Applikation wird über ein Konfigurationsdatei mit dem Namen *config.properties* konfiguriert. Es gibt 2 Möglichkeiten, die Applikation über diese Datei zu konfigurieren:

1. Falls die Systemvariable *moass.auth.config* gesetzt wurde und den absoluten Pfad zur Konfigurationsdatei enthält, wird diese über den angegebenen Wert geladen.
2. Falls obige Systemvariable nicht gesetzt wurde, wird im *Classpath* der Webapplikation nach der Datei *config.properties* gesucht.

Die Konfigurationsdatei enthält folgende Parameter:

Personenbezogene Auslösung MOA-SS Signatur
Signatur im E-Government

Name	Beschreibung
MOASSURL	Die absolute URL auf das MOA-SS Webservice. Beispiel: <code>http://localhost/moa-spss/services/SignatureCreation</code>
KeyIdentifier	Name des Signaturschlüssels, mit welchem die Signatur ausgelöst werden soll. Beispiel: <i>demo</i>

4 Bibliography

Österreich, B. (2006). *MOA: Identifikation (ID)*. Von
<http://www.cio.gv.at/onlineservices/basicmodules/moa-id/> abgerufen

Österreich, B. (2006). *MOA: Serversignatur (SS)*. Von
<http://www.cio.gv.at/onlineservices/basicmodules/moa-spss/> abgerufen

Anhang A

Interface RequestAuthenticator.java

```
package at.gv.egovernment.moa.spss.server.invoke;

import java.util.Hashtable;
import java.util.Set;
import javax.servlet.http.HttpServletRequest;

/**
 * This interface defines the methods for an HTTP based
 * authentication module. Additional parameters for the
 * module are passed using the setParameters
 * method.<p/>
 *
 * Authentication is done by passing the complete HTTP
 * request and the set of accepted roles to the
 * authenticate method. If this module does
 * not operate on role based authentication, this parameter
 * can be ignored.<p/>
 *
 * To decide whether a request can be authenticated by a
 * given module, each authenticator must provide a
 * authenticate method.
 *
 * @author <a href="mailto:arne.tauber@egiz.gv.at">Arne Tauber</a>
 */
public interface RequestAuthenticator {

    /**
     * Sets the parameters that will be passed to the module
     * upon initialization.
     *
     * @param params a hash table containing all the parameter
     * key value pairs.
     */
    void setParameters(Hashtable params);

    /**
     * Authenticates the HTTP signature request.
     *
     * @param request the HTTP signature request.
     * @param roles a set containing the list of
     * accepted roles.
     * @return true if the request can be
     * authenticated, false otherwise.
     */
    boolean authenticate(HttpServletRequest request, Set roles);

    /**
     * Determines whether this authentication module is able
     * to authenticate a given HTTP request.
     *
     * @param request the HTTP request that should be
     * authenticated.
     * @return true if the module is able to
     * authenticate the given HTTP request, false
     * otherwise.
     */
    boolean canAuthenticate(HttpServletRequest request);
}
```