Digital Signature Web Service Interface

1 Introduction

This document describes an RPC interface for a centralized digital signature web service that enforces policy controls on who can request signatures for specific transactions. The signature is calculated using a private key owned by the web service for the purpose of producing an “organization” signature. Thus, anyone within the organization authorized to obtain an “organization” signature can obtain it simply by request to the web service.

1.1 Motivation

A digital signature provides:

Authentication:

> A digital signature is unique to the private key used in its creation, and as a result it provides strong authentication of an individual when that individual signs a contract or e-business transaction.

Support for Non-repudiation:

> Once a contract or transaction has been digitally signed, the signer cannot disclaim or "repudiate" the signature after the fact. This means, for example, that both parties to an online purchase are bound to the terms of the deal - and thus that both parties to the transaction are protected from online fraud.

Data integrity:

> Included in a digital signature is protection of the signed data against any accidental or intentional tampering of the data. For example, the value of an online transaction cannot be compromised without detection, once it has been digitally signed.

Most current implementations of digital signatures bind the public key with a specific individual that is responsible for the content of any data signed with the corresponding private key. However, there is a need, especially in the web services paradigm, for signatures that represent "organizations" (not individuals within organizations) and this need is becoming more apparent over time. Distributing the "organization" private key among all end users authorized to use it creates a number of security concerns. It makes sense then to provide a centralized service which applies all "organization" signatures using a private key unique to the organization. Thus, this document describes an RPC interface for a centralized digital signature web service that enforces policy controls on who can request signatures for specific transactions.

2 Terminology

The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*, and *optional* in this document are to be interpreted as described in **[RFC2119]**.

3 General Interface Design Issues

The major design goals of this specification are simplicity, extensibility and efficiency.

3.1 Related Standards

This specification seeks to leverage both existing and emerging web service standards whenever possible. The following are particularly noted as relevant standardization efforts.

3.1.1 Existing Standards

WSDL – Defines how abstract interfaces and their concrete realizations are defined.

SOAP – Defines how to invoke remote interfaces.

UDDI – Defines how web services are published, queried and found using standardized directories.

SSL/TLS – Defines secure transport mechanisms.

URL – Defines URI (includes URL) syntax and encoding

Character set encoding

XML Digital Signatures – Defines how portions of an XML document are digitally signed.

SAML – Defines how authentication and authorization information may be exchanged.

P3P – Defines how a Producer/entity may publish its privacy policy so that a Consumer could enforce End-User privacy preferences.

3.1.2 Emerging Standards

XML Encryption – Defines how to encrypt/decrypt portions of an XML document.

WS-Security – Defines how document level security standards apply to SOAP messages.

XACML – Defines a syntax for expressing authorization rules.

4 Digital Signature Interfaces

Digital signature interfaces define all operations by the Digital Signature Server that produce a digital signature upon request by a Digital Signature Client.

4.1 envelopingSignXmlData

The **envelopingSignXmlData()** operation returns an enveloping XML Digital Signature on the provided XML markup data.

```
envelopingSignXmlDataResponse =
envelopingSignXmlData(toBeSignedXmlData);
```

Where:

`toBeSignedXmlData` is the XML markup data to be signed. It is of type **string**.

`envelopingSignXmlDataResponse` is the XML markup of the enveloping signature on `toBeSignedXmlData` computed by the Digital Signature Server. It is of type **string**.

This operation SHOULD be invoked only if the requester is authorized to request digital signatures from the Digital Signature Server.

4.2 envelopingSign

The **envelopingSign()** operation returns an enveloping XML Digital Signature on the data pointed to by the given URI.

```
envelopingSignResponse = envelopingSign(toBeSignedUri);
```

Where:

`toBeSignedUri` is the URI pointing to the data to be signed. It is of type **string**.

`envelopingSignResponse` is the XML markup of the enveloping signature on the data pointed to by `toBeSignedUri` computed by the Digital Signature Server. It is of type **string**.

This operation SHOULD be invoked only if the requester is authorized to request digital signatures from the Digital Signature Server.

4.3 detachedSign

The **detachedSign()** operation returns a detached XML Digital Signature on the data pointed to by the given URI.

```
detachedSignResponse = detachedSign(toBeSignedUri);
```

Where:

`toBeSignedUri` is the URI pointing to the data to be signed. It is of type **string**.

`detachedSignResponse` is the XML markup of the detached signature on the data pointed to by `toBeSignedUri` computed by the Digital Signature Server. It is of type **string**.

This operation SHOULD be invoked only if the requester is authorized to request digital signatures from the Digital Signature Server.

4.4 envelopedSign

The **envelopedSign()** operation returns an enveloped XML Digital Signature on the data pointed to by the given URI.

```
envelopedSignResponse = envelopedSign(toBeSignedUri,
signaturePosition);
```

Where:

`toBeSignedUri` is the URI pointing to the data to be signed. It is of type **string**. The data being pointed to MUST be XML markup.

`signaturePosition` is the name of an XML element in the resource to be signed to be used as the insertion point for the signature by the Digital Signature Server. If null, the signature is inserted as the first child element under document root. It is of type **string**.

`envelopedSignResponse` is the XML markup of the enveloped signature on the data pointed to by `toBeSignedUri` computed by the Digital Signature Server. It is of type **string**.

This operation SHOULD be invoked only if the requester is authorized to request digital signatures from the Digital Signature Server.

4.5 envelopedSignXmlData

The **envelopedSignXmlData()** operation returns an enveloped XML Digital Signature on the XML markup data in the request.

```
envelopedSignXmlDataResponse =
envelopedSignXmlData(toBeSignedXmlData, signaturePosition);
```

Where:

`toBeSignedXmlData` is the XML markup data to be signed. It is of type **string**.

`signaturePosition` is the name of an XML element in the resource to be signed to be used as the insertion point for the signature by the Digital Signature Server. If null, the signature is inserted as the first child element under document root. It is of type **string**.

`envelopedSignXmlDataResponse` is the XML markup of the enveloped signature on the data pointed to by `toBeSignedUri` computed by the Digital Signature Server. It is of type **string**.

This operation SHOULD be invoked only if the requester is authorized to request digital signatures from the Digital Signature Server.

5 Security

Digital Signature Servers will be exposed to many of the same security issues as other web service systems. For a representative summary of security concerns, refer to the Security and Privacy Considerations document produced by the XML-Based Security Services Oasis TC.

The Digital Signature Web Server MUST only produce a signature upon request from an authorized digital signature requester. Thus, the requester MUST be authenticated and policy controls determining who is authorized MUST be enforced.

5.1 Authentication of Consumer

Digital Signature Server authentication of a requester may be achieved at the transport level through the use of client certificates in conjunction with SSL/TLS.

5.2 Authentication of Digital Signature Server

Since the requester will likely be sending sensitive data to the Digital Signature Server to be signed, the Server should be authenticated before the data is sent. Authentication of the server may be achieved at the transport level through the use of server certificates in conjunction with SSL/TLS.

5.3 Confidentiality & Message Integrity

SSL/TLS may be used to ensure the contents of messages are neither tampered with nor decipherable by an unauthorized third party.

For Digital Signature Server - requester communications, the use of SSL/TLS is provided by the Server's WSDL declaring an https: entrypoint.

5.4 Access control

A Digital Signature Server MUST implement access control mechanisms that restrict which end entities are authorized to request digital signatures on documents.

6 WSDL Interface Definition

```
<?xml version="1.0" encoding="utf-8" ?>
 <definitions xmlns:s="http://www.w3.org/2001/XMLSchema"
         xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
         xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
         xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:tns="urn:DigSig"
         targetNamespace="urn:DigSig"
         xmlns="http://schemas.xmlsoap.org/wsdl/">

 <message name="envelopingSignXmlData">
  <part name="toBeSignedXmlData"
```

```
        xmlns:partns="http://www.w3.org/2001/XMLSchema"
        type="partns:string" />
</message>

<message name="envelopingSignXmlDataResponse">
 <part name="result"
        xmlns:partns="http://www.w3.org/2001/XMLSchema"
        type="partns:string" />
</message>

<message name="envelopingSign">
 <part name="toBeSignedUri"
        xmlns:partns="http://www.w3.org/2001/XMLSchema"
        type="partns:string" />
</message>

<message name="envelopingSignResponse">
 <part name="result"
        xmlns:partns="http://www.w3.org/2001/XMLSchema"
        type="partns:string" />
</message>

<message name="detachedSign">
 <part name="toBeSignedUri"
        xmlns:partns="http://www.w3.org/2001/XMLSchema"
        type="partns:string" />
</message>

<message name="detachedSignResponse">
 <part name="result"
        xmlns:partns="http://www.w3.org/2001/XMLSchema"
        type="partns:string" />
</message>

<message name="envelopedSign">
 <part name="toBeSignedUri"
        xmlns:partns="http://www.w3.org/2001/XMLSchema"
        type="partns:string" />
<part name="signaturePosition"
        xmlns:partns="http://www.w3.org/2001/XMLSchema"
        type="partns:string" />
</message>

<message name="envelopedSignResponse">
 <part name="result"
        xmlns:partns="http://www.w3.org/2001/XMLSchema"
        type="partns:string" />
</message>

<message name="envelopedSignXmlData">
 <part name="toBeSignedXmlData"
        xmlns:partns="http://www.w3.org/2001/XMLSchema"
        type="partns:string" />
 <part name="signaturePosition"
        xmlns:partns="http://www.w3.org/2001/XMLSchema"
        type="partns:string" />
</message>

<message name="envelopedSignXmlDataResponse">
 <part name="result"
        xmlns:partns="http://www.w3.org/2001/XMLSchema"
        type="partns:string" />
</message>
```

```
<portType name="DigSigPortType">

 <operation name="envelopingSignXmlData">
  <input message="tns:envelopingSignXmlData" />
  <output message="tns:envelopingSignXmlDataResponse" />
 </operation>

 <operation name="envelopingSign">
  <input message="tns:envelopingSign" />
  <output message="tns:envelopingSignResponse" />
 </operation>

 <operation name="detachedSign">
  <input message="tns:detachedSign" />
  <output message="tns:detachedSignResponse" />
 </operation>

 <operation name="envelopedSign">
  <input message="tns:envelopedSign" />
  <output message="tns:envelopedSignResponse" />
 </operation>

 <operation name="envelopedSignXmlData">
  <input message="tns:envelopedSignXmlData" />
  <output message="tns:envelopedSignXmlDataResponse" />
 </operation>

</portType>

<binding name="DigSigSoapBinding" type="tns:DigSigPortType">
 <soap:binding style="rpc"
        transport="http://schemas.xmlsoap.org/soap/http" />

 <operation name="envelopedSignXmlData">
  <soap:operation soapAction="" />
  <input>
   <soap:body use="encoded" namespace="urn:Entrust-DigSig"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        />
  </input>
  <output>
   <soap:body use="encoded" namespace="urn:Entrust-DigSig"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        />
  </output>
 </operation>

 <operation name="envelopingSignXmlData">
  <soap:operation soapAction="" />
  <input>
   <soap:body use="encoded" namespace="urn:Entrust-DigSig"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        />
  </input>
  <output>
   <soap:body use="encoded" namespace="urn:Entrust-DigSig"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        />
  </output>
 </operation>

 <operation name="envelopingSign">
```

```
    <soap:operation soapAction="" />
    <input>
     <soap:body use="encoded" namespace="urn:Entrust-DigSig"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          />
    </input>
    <output>
     <soap:body use="encoded" namespace="urn:Entrust-DigSig"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          />
    </output>
   </operation>

   <operation name="detachedSign">
    <soap:operation soapAction="" />
    <input>
     <soap:body use="encoded" namespace="urn:Entrust-DigSig"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          />
    </input>
    <output>
     <soap:body use="encoded" namespace="urn:Entrust-DigSig"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          />
    </output>
   </operation>

   <operation name="envelopedSign">
    <soap:operation soapAction="" />
    <input>
     <soap:body use="encoded" namespace="urn:Entrust-DigSig"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          />
    </input>
    <output>
     <soap:body use="encoded" namespace="urn:Entrust-DigSig"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          />
    </output>
   </operation>
  </binding>

  <service name="DigSig">
   <documentation>todo: add your documentation here</documentation>
   <port name="DigSigPort" binding="tns:DigSigSoapBinding">
    <soap:address
          location="http://my.org/DigSigServer" />
   </port>
  </service>
</definitions>
```

7 References

7.1 Normative

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

PARTICULAR PURPOSE.