           Using the Elliptic Curve Signature Algorithm (ECDSA)
                       for XML Digital Signatures

Status of This Memo

Copyright Notice

IESG Note

   This document is not a candidate for any level of Internet Standard.
   The IETF disclaims any knowledge of the fitness of this document for
   any purpose, and in particular notes that it has not had IETF review
   for such things as security, congestion control, or inappropriate
   interaction with deployed protocols. The RFC Editor has chosen to
   publish this document at its discretion. Readers of this document
   should exercise caution in evaluating its value for implementation
   and deployment.

Abstract

   This document specifies how to use Elliptic Curve Digital Signature
   Algorithm (ECDSA) with XML Signatures. The mechanism specified
   provides integrity, message authentication, and/or signer
   authentication services for data of any type, whether located within
   the XML that includes the signature or included by reference.

Table of Contents

1. Introduction

   This document specifies how to use the Elliptic Curve Digital
   Signature Algorithm (ECDSA) with XML signatures, as specified in
   [XMLDSIG]. [XMLDSIG] defines only two digital signature methods: RSA
   and DSA (DSS) signatures. This document introduces ECDSA signatures
   as an additional method.

   This document uses both XML Schemas [XML-schema] (normative) and DTDs
   [XML] (informational) to specify the corresponding XML structures.

2. ECDSA

   The Elliptic Curve Digital Signature Algorithm (ECDSA) is the
   elliptic curve analogue of the DSA (DSS) signature method
   [FIPS-186-2]. It is defined in the ANSI X9.62 standard [X9.62].
   Other compatible specifications include FIPS 186-2 [FIPS-186-2], IEEE
   1363 [IEEE1363], IEEE 1363a [IEEE1363a], and SEC1 [SEC1]. [RFC3279]
   describes ways to carry ECDSA keys in X.509 certificates.
   [FIPS-186-2], [SEC2], and [X9.62] provide recommended elliptic curve
   domain parameters for use with ECDSA.

Like DSA, ECDSA incorporates the use of a hash function. Currently,
the only hash function defined for use with ECDSA is the SHA-1
message digest algorithm [FIPS-180-1].

ECDSA signatures are smaller than RSA signatures of similar
cryptographic strength. ECDSA public keys (and certificates) are
smaller than similar strength DSA keys and result in improved
communication efficiency. On many platforms, ECDSA operations can be
computed faster than similar strength RSA or DSA operations (see
[KEYS] for a security analysis of key sizes across public key
algorithms). These advantages of signature size, bandwidth, and
computational efficiency may make ECDSA an attractive choice for
XMLDSIG implementations.

3. Specifying ECDSA within XMLDSIG

This section specifies how to use ECDSA with XML Signature Syntax and
Processing [XMLDSIG]. It relies heavily on the syntax and namespace
defined in [XMLDSIG].

3.1. Version, Namespaces, and Identifiers

This specification makes no provision for an explicit version number
in the syntax. If a future version is needed, it will use a
different namespace.

The XML namespace [XML-ns] URI that MUST be used by implementations
of this (dated) specification is:

    http://www.w3.org/2001/04/xmldsig-more#

Elements in the namespace of the [XMLDSIG] specification are marked
by using the namespace prefix "dsig" in the remaining sections of
this document.

The identifier for the ECDSA signature algorithm as defined in
[Eastlake] is:

    http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha1

3.2. XML Schema Preamble and DTD Replacement

3.2.1. XML Schema Preamble

   The subsequent preamble is to be used with the XML Schema definitions
   given in the remaining sections of this document.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="http://www.w3.org/2001/04/xmldsig-more#"
  xmlns:ecdsa="http://www.w3.org/2001/04/xmldsig-more#"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="0.2">
```

3.2.2. DTD Replacement

   To include ECDSA in XML-signature syntax, the following definition of
   the entity Key.ANY SHOULD replace the one in [XMLDSIG]:

```
<!ENTITY % KeyValue.ANY '| ecdsa:ECDSAKeyValue'>
```

3.3. ECDSA Signatures

   The input to the ECDSA algorithm is the canonicalized representation
   of the dsig:SignedInfo element as specified in Section 3 of
   [XMLDSIG].

   The output of the ECDSA algorithm consists of a pair of integers
   usually referred by the pair (r, s). The signature value (text value
   of element dsig:SignatureValue - see section 4.2 of [XMLDSIG])
   consists of the base64 encoding of the concatenation of two octet-
   streams that respectively result from the octet-encoding of the
   values r and s. This concatenation is described in section E3.1 of
   [IEEE1363].

3.4. ECDSA Key Values

   The syntax used for ECDSA key values closely follows the ASN.1 syntax
   defined in ANSI X9.62 [X9.62].

3.4.1. Key Value Root Element

   The element ECDSAKeyValue is used for encoding ECDSA public keys.
   For use with XMLDSIG, simply use this element inside dsig:KeyValue
   (such as the predefined elements dsig:RSAKeyValue or
   dsig:DSAKeyValue).

   The element consists of an optional subelement DomainParameters and
   the mandatory subelement PublicKey. If Domainparameters is missing,
   the application implicitly knows about it from other means.

Schema Definition:

```
<xs:element name="ECDSAKeyValue" type="ecdsa:ECDSAKeyValueType"/>
<xs:complexType name="ECDSAKeyValueType">
  <xs:sequence>
    <xs:element name="DomainParameters" type="ecdsa:DomainParamsType"
                minOccurs="0"/>
    <xs:element name="PublicKey" type="ecdsa:ECPointType"/>
  </xs:sequence>
</xs:complexType>
```

DTD Definition:

```
<!ELEMENT ECDSAKeyValue (DomainParameters?, PublicKey)>
<!ELEMENT PublicKey (X, Y)?>
<!ELEMENT X EMPTY>
<!ATTLIST X Value CDATA #REQUIRED>
<!ELEMENT Y EMPTY>
<!ATTLIST Y Value CDATA #REQUIRED>
```

3.4.2. EC Domain Parameters

   Domain parameters can be encoded either explicitly using element
   ExplicitParams, or by reference using element NamedCurve. The latter
   simply consists of an attribute named URN, which bears a uniform
   resource name as its value. For the named curves of standards like
   [X9.62], [FIPS-186-2], or [SEC2], the OIDs of these curves SHOULD be
   used in this attribute, e.g., URN="urn:oid:1.2.840.10045.3.1.1". The
   mechanism for encoding OIDs in URNs is shown in [RFC3061].

   Schema Definition:

```
<xs:complexType name="DomainParamsType">
  <xs:choice>
    <xs:element name="ExplicitParams"
                type="ecdsa:ExplicitParamsType"/>
    <xs:element name="NamedCurve">
      <xs:complexType>
        <xs:attribute name="URN" type="xs:anyURI" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:choice>
</xs:complexType>
```

DTD Definition:

```
<!ELEMENT DomainParameters (ExplicitParams | NamedCurve)>
<!ELEMENT NamedCurve EMPTY>
<!ATTLIST NamedCurve URN CDATA #REQUIRED>
```

The element ExplicitParams is used for explicit encoding of domain parameters. It contains three subelements: FieldParams describes the underlying field, CurveParams describes the elliptic curve, and BasePointParams describes the base point of the elliptic curve.

Schema Definition:

```
<xs:complexType name="ExplicitParamsType">
  <xs:sequence>
    <xs:element name="FieldParams" type="ecdsa:FieldParamsType"/>
    <xs:element name="CurveParams" type="ecdsa:CurveParamsType"/>
    <xs:element name="BasePointParams"
                type="ecdsa:BasePointParamsType"/>
  </xs:sequence>
</xs:complexType>
```

DTD Definition:

```
<!ELEMENT ExplicitParams (FieldParams, CurveParams, BasePointParams)>
```

3.4.2.1. Field Parameters

The element FieldParams is used for encoding field parameters. The corresponding XML Schema type FieldParamsType is declared abstract and will be extended by specialized types for prime field, characteristic two field, and odd characteristic extension fields parameters.

The XML Schema type PrimeFieldParamsType is derived from the FieldParamsType and is used for encoding prime field parameters. The type contains the order of the prime field as its single subelement P.

The XML Schema type CharTwoFieldParamsType is derived from FieldParamsType and is used for encoding parameters of a characteristic two field. It is again an abstract type and will be extended by specialized types for trinomial and pentanomial base fields. F2m Gaussian Normal Base fields are not supported by this specification to relieve interoperability. Common to both specialized types is the element M, the extension degree of the field.

The XML Schema type TnBFieldParamsType is derived from the
CharTwoFieldParamsType and is used for encoding trinomial base
fields. It adds the single element K, which represents the integer
k, where x^m + x^k + 1 is the reduction polynomial.

The XML Schema type PnBFieldParamsType is derived from the
CharTwoFieldParamsType and is used for encoding pentanomial base
fields. It adds the three elements K1, K2 and K3, which represent
the integers k1, k2 and k3 respectively, where x^m + x^k3 + x^k2 +
x^k1 + 1 is the reduction polynomial.

The XML Schema type OddCharExtensionFieldParamsType is derived from
the FieldParamsType and is used for encoding parameters of an odd
characteristic extension field. This type contains two elements: M,
which represents the extension degree of the field m, and W, which
represents the integer w, where x^m - w is the reduction polynomial.

Schema Definition:

```
<xs:complexType name="FieldParamsType" abstract="true"/>

<xs:complexType name="PrimeFieldParamsType">
  <xs:complexContent>
    <xs:extension base="ecdsa:FieldParamsType">
      <xs:sequence>
        <xs:element name="P" type="xs:positiveInteger"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="CharTwoFieldParamsType" abstract="true">
  <xs:complexContent>
    <xs:extension base="ecdsa:FieldParamsType">
      <xs:sequence>
        <xs:element name="M" type="xs:positiveInteger"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="OddCharExtensionFieldParamsType">
  <xs:complexContent>
    <xs:extension base="ecdsa:FieldParamsType">
      <xs:sequence>
        <xs:element name="M" type="xs:positiveInteger"/>
        <xs:element name="W" type="xs:positiveInteger"/>
      </xs:sequence>
```

```
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>

      <xs:complexType name="TnBFieldParamsType">
        <xs:complexContent>
          <xs:extension base="ecdsa:CharTwoFieldParamsType">
            <xs:sequence>
              <xs:element name="K" type="xs:positiveInteger"/>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>

      <xs:complexType name="PnBFieldParamsType">
        <xs:complexContent>
          <xs:extension base="ecdsa:CharTwoFieldParamsType">
            <xs:sequence>
              <xs:element name="K1" type="xs:positiveInteger"/>
              <xs:element name="K2" type="xs:positiveInteger"/>
              <xs:element name="K3" type="xs:positiveInteger"/>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
```

DTD Definition:

```
   <!ELEMENT FieldParams (P | (M, K) | (M, K1, K2, K3) | (M, W))>
   <!ELEMENT P (#PCDATA)>
   <!ELEMENT M (#PCDATA)>
   <!ELEMENT K (#PCDATA)>
   <!ELEMENT K1 (#PCDATA)>
   <!ELEMENT K2 (#PCDATA)>
   <!ELEMENT K3 (#PCDATA)>
   <!ELEMENT W (#PCDATA)>
```

3.4.2.2. Curve Parameters

   The element CurveParams is used for encoding parameters of the
   elliptic curve. The corresponding XML Schema type CurveParamsType
   bears the elements A and B representing the coefficients a and b of
   the elliptic curve. According to the algorithm specified in annex
   A3.3 of [X9.62], the optional element Seed contains the value used to
   derive the coefficients of a randomly generated elliptic curve.

Schema Definition:

```
<xs:complexType name="CurveParamsType">
  <xs:sequence>
    <xs:element name="A" type="ecdsa:FieldElemType"/>
    <xs:element name="B" type="ecdsa:FieldElemType"/>
    <xs:element name="Seed" type="xs:hexBinary" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

DTD Definition:

```
<!ELEMENT CurveParams (A, B, Seed?)>
<!ELEMENT A EMPTY>
<!ATTLIST A Value CDATA #REQUIRED>
<!ELEMENT B EMPTY>
<!ATTLIST B Value CDATA #REQUIRED>
<!ELEMENT Seed (#PCDATA)>
```

3.4.2.3. Base Point Parameters

The element BasePointParams is used for encoding parameters regarding
the base point of the elliptic curve. BasePoint represents the base
point itself, Order provides the order of the base point, and
Cofactor optionally provides the cofactor of the base point.

Schema Definition:

```
<xs:complexType name="BasePointParamsType">
  <xs:sequence>
    <xs:element name="BasePoint" type="ecdsa:ECPointType"/>
    <xs:element name="Order" type="xs:positiveInteger"/>
    <xs:element name="Cofactor" type="xs:positiveInteger"
               minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

DTD Definition:

```
<!ELEMENT BasePointParams (BasePoint, Order, Cofactor?)>
<!ELEMENT BasePoint (X, Y)?>
<!ELEMENT Order (#PCDATA)>
<!ELEMENT Cofactor (#PCDATA)>
```

3.4.3. EC Points

   The XML Schema type ECPointType is used for encoding a point on the
   elliptic curve. It consists of the subelements X and Y, providing
   the x and y coordinates of the point. Point compression
   representation is not supported by this specification for the sake of
   simple design.

   The point at infinity is encoded by omitting both elements X and Y.

   The subelements X and Y are of type FieldElemType. This is an
   abstract type for encoding elements of the elliptic curves underlying
   field and is extended by specialized types for prime field elements
   and characteristic two field elements.

   The XML Schema type PrimeFieldElemType is used for encoding prime
   field elements. It contains a single attribute named Value whose
   value represents the field element as an integer.

   The XML Schema type CharTwoFieldElemType is used for encoding
   characteristic two field elements. It contains a single attribute
   named Value whose value represents the field element as an octet
   string. The octet string must be composed as shown in paragraph 2 of
   section 4.3.3 of [X9.62].

   The XML Schema type OddCharExtensionFieldElemType is used for
   encoding odd characteristic extension field elements. It contains a
   single attribute named Value whose value represents the field element
   as an integer. The integer must be composed as shown in section
   5.3.3 of [IEEE1363a].

   Schema Definition:

      <xs:complexType name="ECPointType">
        <xs:sequence minOccurs="0">
          <xs:element name="X" type="ecdsa:FieldElemType"/>
          <xs:element name="Y" type="ecdsa:FieldElemType"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="FieldElemType" abstract="true"/>

```
        <xs:complexType name="PrimeFieldElemType">
          <xs:complexContent>
            <xs:extension base="ecdsa:FieldElemType">
              <xs:attribute name="Value" type="xs:nonNegativeInteger"
                            use="required"/>
            </xs:extension>
          </xs:complexContent>
        </xs:complexType>

        <xs:complexType name="CharTwoFieldElemType">
          <xs:complexContent>
            <xs:extension base="ecdsa:FieldElemType">
              <xs:attribute name="Value" type="xs:hexBinary"
                            use="required"/>
            </xs:extension>
          </xs:complexContent>
        </xs:complexType>

        <xs:complexType name="OddCharExtensionFieldElemType">
          <xs:complexContent>
            <xs:extension base="ecdsa:FieldElemType">
              <xs:attribute name="Value" type="xs:nonNegativeInteger"
                            use="required"/>
            </xs:extension>
          </xs:complexContent>
        </xs:complexType>
```

4. Security Considerations

    Implementers should ensure that appropriate security measures are in
    place when they deploy ECDSA within XMLDSIG. In particular, the
    security of ECDSA requires careful selection of both key sizes and
    elliptic curve domain parameters. Selection guidelines for these
    parameters and some specific recommended curves that are considered
    safe are provided in [X9.62], [FIPS-186-2], and [SEC2]. For further
    security discussion, see [XMLDSIG].

5. Normative References

    [Eastlake]   Eastlake 3rd, D., "Additional XML Security Uniform
                  Resource Identifiers (URIs)", RFC 4051, April 2005.

    [X9.62]   American National Standards Institute. ANSI X9.62-1998,
                  Public Key Cryptography for the Financial Services
                  Industry: The Elliptic Curve Digital Signature
                  Algorithm. January 1999.

    [XMLDSIG]   Eastlake 3rd, D., Reagle, J., and D. Solo, "(Extensible
                  Markup Language) XML-Signature Syntax and Processing",
                  RFC 3275, March 2002.

    [XML-schema] Beech, D., Maloney, M., Mendelsohn, N., and Thompson,
                  H., XML Schema Part 1: Structures, W3C Recommendation,
                  May 2001. http://www.w3.org/TR/2001/REC-xmlschema-1-
                  20010502/ Biron, P., and Malhotra, A., ML Schema Part 2:
                  Datatypes, W3C Recommendation, May 2001.
                  http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

6. Informative References

    [FIPS-180-1] Federal Information Processing Standards Publication
                  (FIPS PUB) 180-1, Secure Hash Standard, April 1995.

    [FIPS-186-2] Federal Information Processing Standards Publication
                  (FIPS PUB) 186-2, Digital Signature Standard, January
                  2000.

    [IEEE1363]   Institute for Electrical and Electronics Engineers
                  (IEEE) Standard 1363-2000, Standard Specifications for
                  Public Key Cryptography, January 2000.

    [IEEE1363a]   Institute for Electrical and Electronics Engineers
                  (IEEE) Standard 1363, Draft Standard Specifications for
                  Public Key Cryptography -- Amendment 1: Additional
                  Techniques, October 2002.

    [KEYS]   Lenstra, A.K. and Verheul, E.R., Selecting Cryptographic
                  Key Sizes. October 1999. Presented at Public Key
                  Cryptography Conference, Melbourne, Australia, January
                  2000. http://www.cryptosavvy.com/

    [RFC3061]   Mealling, M., "A URN Namespace of Object Identifiers",
                  RFC 3061, February 2001.

    [RFC3279]   Bassham, L., Polk, W., and R. Housley, "Algorithms and
                  Identifiers for the Internet X.509 Public Key
                  Infrastructure Certificate and Certificate Revocation
                  List (CRL) Profile", RFC 3279, April 2002.

    [SEC1]   Standards for Efficient Cryptography Group, SEC 1:
                  Elliptic Curve Cryptography, Version 1.0, September
                  2000. http://www.secg.org

   [SEC2]   Standards for Efficient Cryptography Group, SEC 2:
                   Recommended Elliptic Curve Domain Parameters, Version
                   1.0, September 2000.   http://www.secg.org

   [XML]   Bray, T., Maler, E., Paoli, J., and Sperberg-McQueen, C.
                   M., Extensible Markup Language (XML) 1.0 (Second
                   Edition), W3C Recommendation, October 2000.
                   http://www.w3.org/TR/2000/REC-xml-20001006

   [XML-ns]   Bray, T., Hollander, D., and Layman, A., Namespaces in
                   XML, W3C Recommendation, January 1999.
                   http://www.w3.org/TR/1999/REC-xml-names-19990114/

7. Acknowledgements

Appendix A. Aggregate XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="http://www.w3.org/2001/04/xmldsig-more#"
  xmlns:ecdsa="http://www.w3.org/2001/04/xmldsig-more#"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="0.2">

  <!--ECDSA key value root element-->

  <xs:element name="ECDSAKeyValue" type="ecdsa:ECDSAKeyValueType"/>
  <xs:complexType name="ECDSAKeyValueType">
    <xs:sequence>
      <xs:element name="DomainParameters"
                  type="ecdsa:DomainParamsType" minOccurs="0"/>
      <xs:element name="PublicKey" type="ecdsa:ECPointType"/>
    </xs:sequence>
  </xs:complexType>

  <!--EC domain parameters-->

  <xs:complexType name="DomainParamsType">
    <xs:choice>
      <xs:element name="ExplicitParams"
                  type="ecdsa:ExplicitParamsType"/>
      <xs:element name="NamedCurve">
        <xs:complexType>
          <xs:attribute name="URN" type="xs:anyURI" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="FieldParamsType" abstract="true"/>

  <xs:complexType name="PrimeFieldParamsType">
    <xs:complexContent>
      <xs:extension base="ecdsa:FieldParamsType">
        <xs:sequence>
          <xs:element name="P" type="xs:positiveInteger"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="CharTwoFieldParamsType" abstract="true">
    <xs:complexContent>
```

```
        <xs:extension base="ecdsa:FieldParamsType">
          <xs:sequence>
            <xs:element name="M" type="xs:positiveInteger"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="OddCharExtensionFieldParamsType">
      <xs:complexContent>
        <xs:extension base="ecdsa:FieldParamsType">
          <xs:sequence>
            <xs:element name="M" type="xs:positiveInteger"/>
            <xs:element name="W" type="xs:positiveInteger"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="TnBFieldParamsType">
      <xs:complexContent>
        <xs:extension base="ecdsa:CharTwoFieldParamsType">
          <xs:sequence>
            <xs:element name="K" type="xs:positiveInteger"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="PnBFieldParamsType">
      <xs:complexContent>
        <xs:extension base="ecdsa:CharTwoFieldParamsType">
          <xs:sequence>
            <xs:element name="K1" type="xs:positiveInteger"/>
            <xs:element name="K2" type="xs:positiveInteger"/>
            <xs:element name="K3" type="xs:positiveInteger"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ExplicitParamsType">
      <xs:sequence>
        <xs:element name="FieldParams" type="ecdsa:FieldParamsType"/>
        <xs:element name="CurveParams" type="ecdsa:CurveParamsType"/>
        <xs:element name="BasePointParams"
                    type="ecdsa:BasePointParamsType"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="CurveParamsType">
      <xs:sequence>
        <xs:element name="A" type="ecdsa:FieldElemType"/>
```

```
         <xs:element name="B" type="ecdsa:FieldElemType"/>
         <xs:element name="Seed" type="xs:hexBinary" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="BasePointParamsType">
      <xs:sequence>
         <xs:element name="BasePoint" type="ecdsa:ECPointType"/>
         <xs:element name="Order" type="xs:positiveInteger"/>
         <xs:element name="Cofactor" type="xs:positiveInteger"
                    minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>

   <!--EC point-->

   <xs:complexType name="ECPointType">
      <xs:sequence minOccurs="0">
         <xs:element name="X" type="ecdsa:FieldElemType"/>
         <xs:element name="Y" type="ecdsa:FieldElemType"/>
      </xs:sequence>
   </xs:complexType>

   <!--Field element-->

   <xs:complexType name="FieldElemType" abstract="true"/>
   <xs:complexType name="PrimeFieldElemType">
      <xs:complexContent>
         <xs:extension base="ecdsa:FieldElemType">
            <xs:attribute name="Value" type="xs:nonNegativeInteger"
                         use="required"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="CharTwoFieldElemType">
      <xs:complexContent>
         <xs:extension base="ecdsa:FieldElemType">
            <xs:attribute name="Value" type="xs:hexBinary"
                         use="required"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="OddCharExtensionFieldElemType">
      <xs:complexContent>
         <xs:extension base="ecdsa:FieldElemType">
            <xs:attribute name="Value" type="xs:nonNegativeInteger"
                         use="required"/>
         </xs:extension>
      </xs:complexContent>
```

```
      </xs:complexType>
    </xs:schema>
```

Appendix B. Aggregate DTD

```
    <!ELEMENT ECDSAKeyValue (DomainParameters?, PublicKey)>
    <!ELEMENT PublicKey (X, Y)?>
    <!ELEMENT X EMPTY>
    <!ATTLIST X Value CDATA #REQUIRED>
    <!ELEMENT Y EMPTY>
    <!ATTLIST Y Value CDATA #REQUIRED>
    <!ELEMENT DomainParameters (ExplicitParams | NamedCurve)>
    <!ELEMENT NamedCurve EMPTY>
    <!ATTLIST NamedCurve URN CDATA #REQUIRED>
    <!ELEMENT ExplicitParams (FieldParams, CurveParams, BasePointParams)>
    <!ELEMENT FieldParams (P | (M, K) | (M, K1, K2, K3) | (M, W))>
    <!ELEMENT P (#PCDATA)>
    <!ELEMENT M (#PCDATA)>
    <!ELEMENT W (#PCDATA)>
    <!ELEMENT K (#PCDATA)>
    <!ELEMENT K1 (#PCDATA)>
    <!ELEMENT K2 (#PCDATA)>
    <!ELEMENT K3 (#PCDATA)>
    <!ELEMENT CurveParams (A, B, Seed?)>
    <!ELEMENT A EMPTY>
    <!ATTLIST A Value CDATA #REQUIRED>
    <!ELEMENT B EMPTY>
    <!ATTLIST B Value CDATA #REQUIRED>
    <!ELEMENT Seed (#PCDATA)>
    <!ELEMENT BasePointParams (BasePoint, Order, Cofactor?)>
    <!ELEMENT BasePoint (X, Y)?>
    <!ELEMENT Order (#PCDATA)>
    <!ELEMENT Cofactor (#PCDATA)>
```

Authors' Addresses

    Simon Blake-Wilson
    BCI
    96 Spadina Ave, Unit 606
    Toronto, ON, M5V 2J6, Canada

    EMail: sblakewilson@bcisse.com

    Gregor Karlinger
    Federal Staff Office for IT Strategies/Federal Chancellery
    Ballhausplatz 2
    1014 Wien, Austria

    EMail: gregor.karlinger@cio.gv.at

    Tetsutaro Kobayashi
    NTT Laboratories
    1-1 Hikarinooka, Yokosuka, 239-0847, Japan

    EMail: kotetsu@isl.ntt.co.jp

    Yongge Wang
    University of North Carolina at Charlotte
    9201 University City Blvd
    Charlotte, NC 28223, USA

    EMail: yonwang@uncc.edu

Full Copyright Statement

   Copyright (C) The Internet Society (2005).

   This document is subject to the rights, licenses and restrictions
   contained in BCP 78, and except as set forth therein, the authors
   retain all their rights.

   This document and the information contained herein are provided on an
   "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS
   OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET
   ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED,
   INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
   INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
   WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

   The IETF takes no position regarding the validity or scope of any
   Intellectual Property Rights or other rights that might be claimed to
   pertain to the implementation or use of the technology described in
   this document or the extent to which any license under such rights
   might or might not be available; nor does it represent that it has
   made any independent effort to identify any such rights. Information
   on the procedures with respect to rights in RFC documents can be
   found in BCP 78 and BCP 79.

   Copies of IPR disclosures made to the IETF Secretariat and any
   assurances of licenses to be made available, or the result of an
   attempt made to obtain a general license or permission for the use of
   such proprietary rights by implementers or users of this
   specification can be obtained from the IETF on-line IPR repository at
   http://www.ietf.org/ipr.

   The IETF invites any interested party to bring to its attention any
   copyrights, patents or patent applications, or other proprietary
   rights that may cover technology that may be required to implement
   this standard. Please address the information to the IETF at ietf-
   ipr@ietf.org.

Acknowledgement

   Funding for the RFC Editor function is currently provided by the
   Internet Society.