S. Blake-Wilson, BCI
G. Karlinger, CIO Austria
Y. Wang, UNCC

INTERNET-DRAFT
Expires: 31 December 2002

30 June 2002

ECDSA with XML-Signature Syntax
<draft-blake-wilson-xmldsig-ecdsa-03.txt>

Status of this Memo

Abstract

This document specifies how to use ECDSA (Elliptic Curve Digital Signature Algorithm) with XML Signatures [XMLDSIG]. 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]. Therein only two digital signature methods are defined: RSA signatures 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) for specifying the corresponding XML structures.

2. ECDSA

The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analogue of the DSA (DSS) signature method [FIPS186-2]. It is defined in the ANSI X9.62 standard [X9.62]. Other compatible specifications include FIPS 186-2 [FIPS186-2], IEEE 1363 [IEEE1363], and SEC1 [SEC1]. [RFC3279] describes the means to carry ECDSA keys in X.509 certificates. Recommended elliptic curve domain parameters for use with ECDSA are given in [FIPS186-2], [SEC2], and [X9.62].

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, resulting in improved communications

efficiency. Furthermore, 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 the details of how to use ECDSA with XML Signature Syntax and Processing [XMLDSIG]. It relies heavily on the syntax and namespace defined therein.

3.1 Version, Namespaces and Identifiers

No provision is made for an explicit version number in this 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.buergerkarte.at/namespaces/ecdsa/200206030#

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

The identifier for the ECDSA signature algorithm is:
http://www.buergerkarte.at/namespaces/ecdsa/200206030#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.buergerkarte.at/namespaces/
                   ecdsa/200206030#"
  xmlns:ecdsa="http://www.buergerkarte.at/namespaces/ecdsa/200206030#"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
```

```
    version="0.2">
```

3.2.2 DTD Replacement

In order 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 represenation 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 in an instance, this means that the application knows about them from other means (implicitly).

  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 ressource 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 an characteristic two field parameters.

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

The XML Schema type CharTwoFieldParamsType is derived from FieldParamsType as well 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 base fields 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 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 CharTwoFieldParamsType as well 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.

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="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))>
  <!ELEMENT P (#PCDATA)>
  <!ELEMENT M (#PCDATA)>
  <!ELEMENT K (#PCDATA)>
  <!ELEMENT K1 (#PCDATA)>
  <!ELEMENT K2 (#PCDATA)>
  <!ELEMENT K3 (#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, while the optional element Seed contains the value used to derive the coefficients of a randomly generated elliptic curve, according to the algorithm specified in annex A3.3 of [X9.62].

```
  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 curveÆs 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].

```
  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>
```

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 the 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], [NIST-ECC], and [SEC2]. For further security discussion, see [XMLDSIG].

5. Intellectual Property Rights

The IETF has been notified of intellectual property rights claimed in

regard to the specification contained in this document.
For more information, consult the online list of claimed rights
(http://www.ietf.org/ipr.html).

The IETF takes no position regarding the validity or scope of any intellectual property 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; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication 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 Secretariat.

6. 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.

[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., RFC 3061, A URN Namespace of Object Identifiers. IETF Informational RFC, February 2001.

http://www.ietf.org/rfc/rfc3061.txt

[RFC3279] Bassham, L., Housley, R., and Polk, W., RFC 3279, Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. IETF Proposed Standard, April 2002. http://www.ietf.org/rfc/rfc3279.txt

[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

[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.

[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

[XMLDSIG] Eastlake, D., Reagle, J., and Solo, D., XML-Signature Syntax and Processing. W3C Recommendation, February 2002. http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/

[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/

[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/

7. Authors' Addresses

Simon Blake-Wilson
BCI
96 Spadina Ave, Unit 606
Toronto, ON, M5V 2J6, Canada
e-mail: sblakewilson@bcisse.com

Gregor Karlinger
Chief Information Office Austria
Parkring 10/I/5
1010 Wien, Austria
e-mail: gregor.karlinger@cio.gv.at

Yongge Wang
University of North Carolina at Charlotte
9201 University City Blvd
Charlotte, NC 28223, USA
e-mail: ywang@uncc.edu

8. Acknowledgements

The authors would like to acknowledge the many helpful comments of Wolfgang Bauer, Donald Eastlake, Tom Gindin, Chris Hawk, Joseph M. Reagle Jr., and Francois Rousseau.

9. Full Copyright Statement

Appendix A: Aggregate XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.buergerkarte.at/namespaces/
                            ecdsa/200206030#"
           xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ecdsa="http://www.buergerkarte.at/namespaces/
                        ecdsa/200206030#"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified" version="0.3">

  <!--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="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:schema>
```

Appendix A: 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))>
  <!ELEMENT P (#PCDATA)>
  <!ELEMENT M (#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)>
```