PKIX Working Group L. Bassham (NIST)
Internet Draft D. Johnson (Certicom)
expires December 3, 1999 W. Polk (NIST)
June 3, 1999

Internet X.509 Public Key Infrastructure

Representation of Elliptic Curve Digital Signature Algorithm (ECDSA) Keys and Signatures in Internet X.509 Public Key Infrastructure Certificates

<draft-ietf-pkix-ipki-ecdsa-01.txt>

Status of this Memo

Abstract

This is the third draft of a profile for specification of Elliptic Curve Digital Signature Algorithm (ECDSA) keys and signatures in Internet Public Key Infrastructure X.509 certificates and certificate revocation lists. This specification is an addendum to RFC 2459; implementations must also conform to RFC 2459. In addition, this document references ANSI X9.62 for the specification of the ECDSA algorithm. This specification only addresses the format of ECDSA keys and signatures. Please send comments on this document to the ietf-pkix@imc.org mail list.

1 Executive Summary

This specification contains guidance on the use of the Internet Public Key Infrastructure certificates to convey Elliptic Curve Digital Signature Algorithm (ECDSA) keys. This specification is an addendum to RFC 2459, "Internet Public Key Infrastructure: X.509 Certificate and CRL Profile". Implementations of this specification must also conform to RFC 2459. Implementations of this specification are not required to conform to other parts from that series.

The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analog of the Digital Signature Algorithm (DSA). This specification profiles the format and semantics of fields in X.509 V3 certificates containing ECDSA keys. The specification addresses the subjectPublicKeyInfo field and the keyUsage extension.

2 Requirements and Assumptions

The goal is to augment the X.509 certificate profile presented in Part 1 to facilitate the use and management of ECDSA keys for those communities which use this algorithm.

2.1 Communication and Topology

This profile, as presented in Part 1 and augmented by this specification, supports users without high bandwidth, real-time IP connectivity, or high connection availability. In addition, the profile allows for the presence of firewall or other filtered communication.

This profile does not assume the deployment of an X.500 Directory system. The profile does not prohibit the use of an X.500 Directory, but other means of distributing certificates and certificate revocation lists (CRLs) are supported.

2.2 Acceptability Criteria

The goal of the Internet Public Key Infrastructure (PKI) is to meet the needs of deterministic, automated identification, authentication, access control, and authorization functions. Support for these services determines the attributes contained in the certificate as well as the ancillary control information in the certificate such as policy data and certification path constraints.

The goal of this document is to profile ECDSA certificates, specifying the contents and semantics of attributes which were not fully specified by Part 1. If not specifically addressed by this document, the contents and semantics of the fields and extensions must be as described in Part 1.

2.3 User Expectations

Users of the Internet PKI are people and processes who use client software and are the subjects named in certificates. These uses include readers and writers of electronic mail, the clients for WWW browsers, WWW servers, and the key manager for IPSEC within a router. This profile recognizes the limitations of the platforms these users employ and the sophistication/attentiveness of the users themselves. This manifests itself in minimal user configuration responsibility (e.g., root keys, rules), explicit platform usage constraints within the certificate, certification path constraints which shield the user from many malicious actions, and applications which sensibly automate validation functions.

2.4 Administrator Expectations

As with users, the Internet PKI profile is structured to support the individuals who generally operate Certification Authorities (CAs). Providing administrators with unbounded choices increases the chances that a subtle CA administrator mistake will result in broad compromise or unnecessarily limit interoperability. This profile defines the object identifiers and data formats that must be supported to interpret ECDSA public keys.

3 ECDSA Algorithm Support

This section describes object identifiers and data formats which may be used with PKIX certificate profile to describe X.509 certificates containing an ECDSA public key or signed with ECDSA. Conforming CAs are required to use the object identifiers and data formats when issuing ECDSA certificates. Conforming applications shall recognize the object identifiers and process the data formats when processing such certificates.

The Elliptic Curve Digital Signature Algorithm (ECDSA) is defined in the ANSI X9.62 standard [X9.62]. The ASN.1 object identifiers used to identify the ECDSA algorithm are defined in the following arc:

```
ansi-X9-62 OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) 10045 }
```

3.1 Subject Public Key Info

The certificate identifies the ECDSA algorithm, conveys optional parameters, and specifies the ECDSA public key in the subjectPublicKeyInfo field. The subjectPublicKeyInfo field is a

SEQUENCE of an algorithm identifier and the subjectPublicKey field.

The certificate indicates the algorithm through an algorithm identifier. This algorithm identifier consists of an object identifier (OID) and optional associated parameters. Section 3.1.1 identifies the preferred OID and parameters for the ECDSA algorithm. Conforming CAs shall use the identified OID when issuing certificates containing public keys for the ECDSA algorithm. Conforming applications supporting the ECDSA algorithm shall, at a minimum, recognize the OID identified in section 3.1.1.

The certificate conveys the ECDSA public key through the subjectPublicKey field. This subjectPublicKey field is a BIT STRING. Section 3.1.2 specifies the method for encoding a ECDSA public key as a BIT STRING. Conforming CAs shall encode the ECDSA public key as described in Section 3.1.2 when issuing certificates containing public keys for the ECDSA algorithm. Conforming applications supporting the ECDSA algorithm shall decode the subjectPublicKey as described in section 3.1.2 when the algorithm identifier is the one presented in 3.1.1.

3.1.1 Algorithm Identifier and Parameters

When certificates contain an ECDSA public key, the id-ecPublicKey algorithm identifier shall be used. The id-ecPublicKey algorithm identifier is defined as follows:

```
id-public-key-type OBJECT IDENTIFIER ::= { ansi-X9.62 2 }

id-ecPublicKey OBJECT IDENTIFIER ::= { id-publicKeyType 1 }
```

ECDSA requires use of certain parameters with the public key. The parameters may be inherited from the issuer, implicitly included through reference to a "named curve," or explicitly included in the certificate.

```
Parameters ::= CHOICE {
ecParameters ECParameters,
namedCurve CURVES.&id({CurveNames}),
implicitlyCA NULL }
```

When the parameters are inherited, the parameters field shall contain implictlyCA, which is the ASN.1 value NULL. When parameters are specified by reference, the parameters field shall contain the namedCurve choice, which is an an object identifier. When the parameters are explicitly included, they shall be encoded in the ASN.1 structure ECParameters:

```
ECParameters ::= SEQUENCE {
version INTEGER { ecpVer1(1) } (ecpVer1),
-- version is always 1
fieldID FieldID { {FieldTypes} },
-- identifies the finite field over
-- which the curve is defined
curve Curve, -- coefficients a and b of the
-- elliptic curve
base ECPoint, -- specifies the base point P
-- on the elliptic curve
order INTEGER, -- the order n of the base point
cofactor INTEGER OPTIONAL,
... }

FieldElement ::= OCTET STRING
```

The value of FieldElement shall be the octet string representation of a field element following the conversion routine in [X9.62, Section 4.3.1]

```
Curve ::= SEQUENCE {
a FieldElement,
b FieldElement,
seed BIT STRING OPTIONAL
}

ECPoint ::= OCTET STRING
```

The value of ECPoint shall be the octet string representation of an elliptic curve point following the conversion routine in [X9.62, Section 4.4.3.b]

The components of type ECParameters have the following meanings:

* version specifies the version number of the elliptic curve parameters. It shall have the value 1 for this version of the Standard. The notation above creates an INTEGER named ecpVer1 and gives it a value of one. It is used to constrain version to a single value.

* fieldID identifies the finite field over which the elliptic curve is defined. Finite fields are represented by values of the parameterized type FieldID, constrained to the values of the objects defined in the information object set FieldTypes. Additional detail regarding fieldID is provided below.

* curve specifies the coefficients a and b of the elliptic curve E. Each coefficient shall be represented as a value of type

FieldElement, an OCTET STRING. seed is an optional parameter used to derive the coefficients of a randomly generated elliptic curve.

* base specifies the base point P on the elliptic curve. The base point shall be represented as a value of type ECPoint, an OCTET STRING.

* order specifies the order n of the base point.

* cofactor is the integer h = #E(Fq)/n. Note: This optional parameter is not used in ECDSA, except in parameter validation. Parameter validation is not required by this specification. It is included for compatibility with Elliptic Curve Key Agreement public key parameters, and to support parameter validation.

The AlgorithmIdentifier within subjectPublicKeyInfo is the only place within a certificate where the parameters may be used. If the ECDSA algorithm parameters are absent from the subjectPublicKeyInfo AlgorithmIdentifier and the CA signed the subject certificate using ECDSA, then the certificate issuer's ECDSA parameters apply to the subject's ECDSA key. If the ECDSA algorithm parameters are absent from the subjectPublicKeyInfo AlgorithmIdentifier and the CA signed the certificate using a signature algorithm other than ECDSA, then clients shall not validate the certificate.

```
FieldID { FIELD-ID:IOSet } ::= SEQUENCE {
fieldType FIELD-ID.&id({IOSet}),
parameters FIELD-ID.&Type({IOSet}{@fieldType}) OPTIONAL
}
FieldTypes FIELD-ID ::= {
{ Prime-p IDENTIFIED BY prime-field } |
{ Characteristic-two IDENTIFIED BY characteristic-two-field },
...
}
FIELD-ID ::= TYPE-IDENTIFIER
```

FieldID is a parameterized type composed of two components, fieldType and parameters. These components are specified by the fields &id and &Type, which form a template for defining sets of information objects, instances of the class FIELD-ID. This class is based on the useful information object class TYPE-IDENTIFIER, described in X.681 Annex A. In an instance of FieldID, "fieldType" will contain an object identifier value that uniquely identifies the type contained in "parameters". The effect of referencing "fieldType" in both components of the fieldID sequence is to tightly bind the object identifier and its type.

The information object set FieldTypes is used as the single parameter in a reference to type FieldID. FieldTypes contains two objects followed by the extension marker ("..."). Each object, which represents a finite field, contains a unique object identifier and its associated type. The values of these objects define all of the valid values that may appear in an instance of fieldID. The extension marker allows backward compatibility with future versions of this standard which may define objects to represent additional kinds of finite fields.

The object identifier id-fieldType represents the root of a tree containing the object identifiers of each field type. It has the following value:

```
id-fieldType OBJECT IDENTIFIER ::= { ansi-X9-62 fieldType(1) }
```

The object identifiers prime-field and characteristic-two-field name the two kinds of fields defined in this Standard. They have the following values:

```
prime-field OBJECT IDENTIFIER ::= { id-fieldType 1 }

characteristic-two-field OBJECT IDENTIFIER ::= { id-fieldType 2 }

Prime-p ::= INTEGER -- Field size p (p in bits)

Characteristic-two ::= SEQUENCE {
m INTEGER, -- Field size 2^m (m in bits)
basis CHARACTERISTIC-TWO.&id({BasisTypes}),
parameters CHARACTERISTIC-TWO.&Type({BasisTypes}{@basis})
}

BasisTypes CHARACTERISTIC-TWO::= {
{ NULL IDENTIFIED BY gnBasis } |
{ Trinomial IDENTIFIED BY tpBasis } |
{ Pentanomial IDENTIFIED BY ppBasis },
...
}

Trinomial ::= INTEGER

Pentanomial ::= SEQUENCE {
k1 INTEGER,
k2 INTEGER,
k3 INTEGER
}

CHARACTERISTIC-TWO ::= TYPE-IDENTIFIER
```

The object identifier id-characteristic-two-basis represents the root of a tree containing the object identifiers for each type of basis for the characteristic-two finite fields. It has the following value:

```
id-characteristic-two-basis OBJECT IDENTIFIER ::= {
characteristic-two-field basisType(1) }
```

The object identifiers gnBasis, tpBasis and ppBasis name the three kinds of basis for characteristic-two finite fields defined by [X9.62]. They have the following values:

```
gnBasis OBJECT IDENTIFIER ::= { id-characteristic-two-basis 1 }
tpBasis OBJECT IDENTIFIER ::= { id-characteristic-two-basis 2 }
ppBasis OBJECT IDENTIFIER ::= { id-characteristic-two-basis 3 }
```

3.1.2 Encoding of ECDSA Public Keys

The elliptic curve public key (an ECPoint which is an OCTET STRING) is mapped to a subjectPublicKey (a BIT STRING) as follows: the most significant bit of the OCTET STRING becomes the most significant bit of the BIT STRING, etc.; the least significant bit of the OCTET STRING becomes the least significant bit of the BIT STRING.

3.1.3 Key Usage Extension in ECDSA certificates

The key usage extension may optionally appear in certificates which convey an ECDSA public key. If a certificate containing an ECDSA public key includes the keyUsage extension, only the following values may be asserted:

digitalSignature;
nonRepudiation;
keyCertSign; and
cRLSign.

The keyCertSign and cRLSign values may only be asserted if the basicConstraints extension is present and cA is TRUE.

3.2 Representation of ECDSA Signatures

When used to sign certificates, CRLs, or PKI messages, the ECDSA shall be used with the SHA-1 hash algorithm. The ASN.1 object identifier used to identify the ECDSA algorithm with SHA-1 shall be:

```
id-ecSigType OBJECT IDENTIFIER ::= { ansi-X9-62 signatures(4) }
ecdsa-with-SHA1 OBJECT IDENTIFIER ::= { id-ecSigType 1 }
```

When the ecdsa-with-SHA1 algorithm identifier is used in the SIGNED parameterized TYPE (e.g., in the signature on a certificate or CRL) it shall have NULL parameters. The ECDSA parameters in the certificate of the issuer shall apply to the verification of the signature. When signing, the ECDSA algorithm generates two values. These values are commonly referred to as r and s. To easily transfer these two values as one signature, they shall be ASN.1 encoded using the following ASN.1 structure:

```
Ecdsa-Sig-Value ::= SEQUENCE {
r INTEGER,
s INTEGER }
```

4 ASN.1 Module

```
ANSI-X9-62 { iso(1) member-body(2) us(840) 10045 module(4) 1 }
DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- EXPORTS All;

-- IMPORTS None;

ansi-X9-62 OBJECT IDENTIFIER ::= {
iso(1) member-body(2) us(840) 10045 }

FieldID { FIELD-ID:IOSet } ::= SEQUENCE { -- Finite field
fieldType FIELD-ID.&id({IOSet}),
parameters FIELD-ID.&Type({IOSet}{@fieldType})
}

FieldTypes FIELD-ID ::= {
{ Prime-p IDENTIFIED BY prime-field } |
{ Characteristic-two IDENTIFIED BY characteristic-two-field },
...
}

FIELD-ID ::= TYPE-IDENTIFIER -- ISO/IEC 8824-2:1995(E), Annex A

id-fieldType OBJECT IDENTIFIER ::= { ansi-X9-62 fieldType(1) }

prime-field OBJECT IDENTIFIER ::= { id-fieldType 1 }

characteristic-two-field OBJECT IDENTIFIER ::= { id-fieldType 2 }

Prime-p ::= INTEGER -- Finite field F(p), where p is an odd prime

Characteristic-two ::= SEQUENCE {
```

```
m INTEGER, -- Field size 2^m
basis CHARACTERISTIC-TWO.&id({BasisTypes}),
parameters CHARACTERISTIC-TWO.&Type({BasisTypes}{@basis})
}

BasisTypes CHARACTERISTIC-TWO::= {
{ NULL IDENTIFIED BY gnBasis } |
{ Trinomial IDENTIFIED BY tpBasis } |
{ Pentanomial IDENTIFIED BY ppBasis },
...
}

-- Trinomial basis representation of F2^m
-- Integer k for reduction polynomial xm + xk + 1
--
Trinomial ::= INTEGER

Pentanomial ::= SEQUENCE {
--
-- Pentanomial basis representation of F2^m
-- reduction polynomial integers k1, k2, k3
-- f(x) = x**m + x**k3 + x**k2 + x**k1 + 1
--
k1 INTEGER,
k2 INTEGER,
k3 INTEGER
}

CHARACTERISTIC-TWO ::= TYPE-IDENTIFIER

id-characteristic-two-basis OBJECT IDENTIFIER ::= {
characteristic-two-field basisType(3) }

-- The object identifiers gnBasis, tpBasis and ppBasis name
-- three kinds of basis for characteristic-two finite fields

gnBasis OBJECT IDENTIFIER ::= { id-characteristic-two-basis 1 }

tpBasis OBJECT IDENTIFIER ::= { id-characteristic-two-basis 2 }

ppBasis OBJECT IDENTIFIER ::= { id-characteristic-two-basis 3 }

FieldElement ::= OCTET STRING -- Finite field element

ECPoint ::= OCTET STRING -- Elliptic curve point

ECParameters ::= SEQUENCE { -- Elliptic curve parameters
version INTEGER { ecpVer1(1) } (ecpVer1),
fieldID FieldID {{FieldTypes}},
curve Curve,
base ECPoint, -- Base point G
order INTEGER, -- Order n of the base point
cofactor INTEGER OPTIONAL, -- The integer h = #E(Fq)/n
...
}

Curve ::= SEQUENCE {
a FieldElement, -- Elliptic curve coefficient a
b FieldElement, -- Elliptic curve coefficient b
seed BIT STRING OPTIONAL
}

ECDSA-Sig-Value ::= SEQUENCE {
r INTEGER,
s INTEGER
}

id-ecSigType OBJECT IDENTIFIER ::= { ansi-X9-62 signatures(4) }

ecdsa-with-SHA1 OBJECT IDENTIFIER ::= { id-ecSigType 1 }

SubjectPublicKeyInfo ::= SEQUENCE {
algorithm AlgorithmIdentifier {{ECPKAlgorithms}},
subjectPublicKey BIT STRING
}

AlgorithmIdentifier { ALGORITHM:IOSet } ::= SEQUENCE {
algorithm ALGORITHM.&id({IOSet}),
parameters ALGORITHM.&Type({IOSet}{@algorithm})
}

ECPKAlgorithms ALGORITHM ::= {
ecPublicKeyType,
...
}

ecPublicKeyType ALGORITHM ::= {
Parameters IDENTIFIED BY id-ecPublicKey
}

ALGORITHM ::= TYPE-IDENTIFIER

id-publicKeyType OBJECT IDENTIFIER ::= { ansi-X9-62 keyType(2) }

id-ecPublicKey OBJECT IDENTIFIER ::= { id-publicKeyType 1 }
```

```
Parameters ::= CHOICE {
ecParameters ECParameters,
namedCurve CURVES.&id({CurveNames}),
implicitlyCA NULL
}

CurveNames CURVES ::= {
{ ID c2pnb163v1 } | -- J.4.1, example 1 --
{ ID c2pnb163v2 } | -- J.4.1, example 2 --
{ ID c2pnb163v3 } | -- J.4.1, example 3 --
{ ID c2pnb176w1 } | -- J.4.2, example 1 --
{ ID c2tnb191v1 } | -- J.4.3, example 1 --
{ ID c2tnb191v2 } | -- J.4.3, example 2 --
{ ID c2tnb191v3 } | -- J.4.3, example 3 --
{ ID c2onb191v4 } | -- J.4.3, example 4 --
{ ID c2onb191v5 } | -- J.4.3, example 5 --
{ ID c2pnb208w1 } | -- J.4.4, example 1 --
{ ID c2tnb239v1 } | -- J.4.5, example 1 --
{ ID c2tnb239v2 } | -- J.4.5, example 2 --
{ ID c2tnb239v3 } | -- J.4.5, example 3 --
{ ID c2onb239v4 } | -- J.4.5, example 4 --
{ ID c2onb239v5 } | -- J.4.5, example 5 --
{ ID c2pnb272w1 } | -- J.4.6, example 1 --
{ ID c2pnb304w1 } | -- J.4.7, example 1 --
{ ID c2tnb359v1 } | -- J.4.8, example 1 --
{ ID c2pnb368w1 } | -- J.4.9, example 1 --
{ ID c2tnb431r1 } | -- J.4.10, example 1 --
{ ID prime192v1 } | -- J.5.1, example 1 --
{ ID prime192v2 } | -- J.5.1, example 2 --
{ ID prime192v3 } | -- J.5.1, example 3 --
{ ID prime239v1 } | -- J.5.2, example 1 --
{ ID prime239v2 } | -- J.5.2, example 2 --
{ ID prime239v3 } | -- J.5.2, example 3 --
{ ID prime256v1 }, -- J.5.3, example 1 --
... -- others --
}

CURVES ::= CLASS {
&id OBJECT IDENTIFIER UNIQUE
}
WITH SYNTAX { ID &id }

ellipticCurve OBJECT IDENTIFIER ::= { ansi-X9-62 curves(3) }

c-TwoCurve OBJECT IDENTIFIER ::= {
ellipticCurve characteristicTwo(0) }

primeCurve OBJECT IDENTIFIER ::= { ellipticCurve prime(1) }
```

```
c2pnb163v1 OBJECT IDENTIFIER ::= { c-TwoCurve 1 }
c2pnb163v2 OBJECT IDENTIFIER ::= { c-TwoCurve 2 }
c2pnb163v3 OBJECT IDENTIFIER ::= { c-TwoCurve 3 }
c2pnb176w1 OBJECT IDENTIFIER ::= { c-TwoCurve 4 }
c2tnb191v1 OBJECT IDENTIFIER ::= { c-TwoCurve 5 }
c2tnb191v2 OBJECT IDENTIFIER ::= { c-TwoCurve 6 }
c2tnb191v3 OBJECT IDENTIFIER ::= { c-TwoCurve 7 }
c2onb191v4 OBJECT IDENTIFIER ::= { c-TwoCurve 8 }
c2onb191v5 OBJECT IDENTIFIER ::= { c-TwoCurve 9 }
c2pnb208w1 OBJECT IDENTIFIER ::= { c-TwoCurve 10 }
c2tnb239v1 OBJECT IDENTIFIER ::= { c-TwoCurve 11 }
c2tnb239v2 OBJECT IDENTIFIER ::= { c-TwoCurve 12 }
c2tnb239v3 OBJECT IDENTIFIER ::= { c-TwoCurve 13 }
c2onb239v4 OBJECT IDENTIFIER ::= { c-TwoCurve 14 }
c2onb239v5 OBJECT IDENTIFIER ::= { c-TwoCurve 15 }
c2pnb272w1 OBJECT IDENTIFIER ::= { c-TwoCurve 16 }
c2pnb304w1 OBJECT IDENTIFIER ::= { c-TwoCurve 17 }
c2tnb359v1 OBJECT IDENTIFIER ::= { c-TwoCurve 18 }
c2pnb368w1 OBJECT IDENTIFIER ::= { c-TwoCurve 19 }
c2tnb431r1 OBJECT IDENTIFIER ::= { c-TwoCurve 20 }

prime192v1 OBJECT IDENTIFIER ::= { primeCurve 1 }
prime192v2 OBJECT IDENTIFIER ::= { primeCurve 2 }
prime192v3 OBJECT IDENTIFIER ::= { primeCurve 3 }
prime239v1 OBJECT IDENTIFIER ::= { primeCurve 4 }
prime239v2 OBJECT IDENTIFIER ::= { primeCurve 5 }
prime239v3 OBJECT IDENTIFIER ::= { primeCurve 6 }
prime256v1 OBJECT IDENTIFIER ::= { primeCurve 7 }

END
```

5 References

[X9.62] X9.62-1999, "Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)".

[P1363] IEEE P1363, "Standard for Public-Key Cryptography", draft standard, 1997.

[RFC 2459] R. Housley, W. Ford, W. Polk and D. Solo "Internet X.509 Public Key Infrastructure: Certificate and CRL Profile", January, 1999.

6 Intellectual Property Rights

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information, consult the online list of claimed rights.

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 implementors or users of this specification can be obtained from the IETF Secretariat.

7 Security Considerations

This specification does not constrain the key sizes or identify particular elliptic curves for use in the Internet PKI. However, both the key size and the particular curve selected impact the the strength of the digital signatures. Some curves are cryptographically stronger than others!

In general, use of "well-known" curves, such as the "named curves" from ANSI X9.62 is a sound strategy. For additional information, refer to X9.62 Appendix D.4, "Key Length Considerations" and Appendix F.1, "Avoiding Cryptographically Weak Keys".

This specification is a profile of RFC 2459. The security considerations section of that document applies to this specification as well.

8 Author Addresses:

Larry Bassham
NIST
100 Bureau Drive, Stop 8930
Gaithersburg, MD 20899-8930
USA
lbassham@nist.gov

Don Johnson

Certicom
4253 Sleepy Lake Drive
Fairfax, VA 22033
USA
djohnson@certicom.com

Tim Polk
NIST
100 Bureau Drive, Stop 8930
Gaithersburg, MD 20899-8930
USA
wpolk@nist.gov