| | |
|---|---|
| Network Working Group | S. Josefsson |
| Internet-Draft | SJD AB |
| Intended status: Standards Track | S. Leonard |
| Expires: March 19, 2015 | Penango, Inc. |
| | September 15, 2014 |

# Text Encodings of PKIX and CMS Structures

**draft-josefsson-pkix-textual-06**

## Abstract

This document describes and discusses the text encodings of the Public-Key Infrastructure X.509 (PKIX), Public-Key Cryptography Standards (PKCS), and Cryptographic Message Syntax (CMS), The text encodings are well-known, are implemented by several applications and libraries, and are widely deployed. This document is intended to articulate the de-facto rules that existing implementations operate by, and to give recommendations that will promote interoperability going forward.

## Table of Contents


1. Introduction
2. General Considerations
3. ABNF
4. Guide
5. Text Encoding of Certificates
5.1. Encoding
5.2. Explanatory Text
5.3. File Extension
6. Text Encoding of Certificate Revocation Lists
7. Text Encoding of PKCS #10 Certification Request Syntax
8. Text Encoding of PKCS #7 Cryptographic Message Syntax
9. Text Encoding of Cryptographic Message Syntax
10. Text Encoding of PKCS #8 Private Key Info, and One Asymmetric Key
11. Text Encoding of PKCS #8 Encrypted Private Key Info
12. Text Encoding of Attribute Certificates
13. Text Encoding of Subject Public Key Info
14. Security Considerations
15. IANA Considerations
16. Acknowledgements
17. References
17.1. Normative References
17.2. Informative References
Appendix A. Non-Conforming Examples
Appendix B. DER Expectations
Authors' Addresses


## 1. Introduction

Several security-related standards used on the Internet define ASN.1 data formats that are normally encoded using the Basic Encoding Rules (BER) or Distinguished Encoding Rules (DER) [X.690], which are binary, octet-oriented encodings. This document is about the text encodings of these formats:

1. Certificates, Certificate Revocation Lists (CRLs), and Subject Public Key Info structures in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile [RFC5280].
2. PKCS #10: Certification Request Syntax [RFC2986].
3. PKCS #7: Cryptographic Message Syntax [RFC2315].
4. Cryptographic Message Syntax [RFC5652].
5. PKCS #8: Private-Key Information Syntax [RFC5208], renamed to One Asymmetric Key in Asymmetric Key Package [RFC5958], and Encrypted Private-Key Information Syntax in the same standards.
6. Attribute Certificates in An Internet Attribute Certificate Profile for Authorization [RFC5755].

Although other formats exist that use the text encoding (or something like it) described in this document, the included formats share a common property: algorithm agility. "Algorithm agility" means that different algorithms to achieve the same purposes—such as content encryption or integrity protection—can be used in different instances of the same format because the instance data identifies the algorithms and associated parameters. Weakness in an algorithm does not destroy the utility of the format.

A disadvantage of a binary data format is that it cannot be interchanged in textual transports, such as e-mail or text documents. One advantage with text encodings is that they are easy to modify using common text editors; for example, a user may concatenate several certificates to form a certificate chain with copy-and-paste operations.

The tradition within the RFC series can be traced back to PEM [RFC1421], based on a proposal by M. Rose in Message Encapsulation [RFC0934]. Originally called "PEM encapsulation mechanism", "encapsulated PEM message", or (arguably) "PEM printable encoding", today the format is sometimes referred to as "PEM encoding". Variations include OpenPGP ASCII Armor [RFC2015] and OpenSSH Key File Format [RFC4716].

For reasons that basically boil down to non-coordination or inattention, many PKIX and CMS libraries implement a text encoding that is similar to—but not identical with—PEM encoding. This document specifies the "PKIX text encoding" format, articulates the de-facto rules that most implementations operate by, and provides recommendations that will promote interoperability going forward. This document also provides common nomenclature for syntax elements, reflecting the evolution of this de-facto standard format. Peter Gutmann's X.509 Style Guide [X.509SG] contains a section "base64 Encoding" that describes the formats and contains suggestions similar to what is in this document.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## 2. General Considerations

PKIX text encoding begins with a line starting with `-----BEGIN` and ends with a line starting with `-----END`. Between these lines, or "encapsulation boundaries", are base64-encoded [RFC4648] data. Data before the `-----BEGIN` and after the `-----END` encapsulation boundaries are permitted and MUST NOT cause parsers to malfunction. Furthermore, parsers MUST ignore whitespace and other non-base64 characters and MUST handle different newline conventions.

The type of data encoded is labeled depending on the type label in the `-----BEGIN` line (pre-encapsulation boundary). For example, the line may be `-----BEGIN CERTIFICATE-----` to indicate that the content is a PKIX certificate (see further below). Generators MUST put the same label on the `-----END` line (post-encapsulation boundary) as the corresponding `-----BEGIN` line. Parsers MAY disregard the label on the `-----END` line instead of signaling an error if there is a label mismatch. There is exactly one space character (`SP`) separating the `BEGIN` or `END` from the label. There are exactly five hyphen-minus (or dash) characters (`"-"`) on both ends of the encapsulation boundaries, no more, no less.

The label type implies that the encoded data follows the specified syntax. Parsers MUST handle non-conforming data gracefully. However, not all parsers or generators prior to this Internet-Draft behave consistently. A conforming parser MAY interpret the contents as another label type, but ought to be aware of the security implications discussed in the Security Considerations section. Consistent with algorithm agility, the labels described in this document are not specific to any particular cryptographic algorithm.

Unlike legacy PEM encoding [RFC1421], OpenPGP ASCII armor, and the OpenSSH key file format, PKIX text encoding does **not** define or permit attributes to be encoded alongside the PKIX or CMS data. Whitespace MAY appear between the pre-encapsulation boundary and the base64, but generators SHOULD NOT emit such whitespace.

Files MAY contain multiple PKIX text encoding instances. This is used, for example, when a file contains several certificates. Whether the instances are ordered or unordered depends on the context.

Generators MUST wrap the base64 encoded lines so that each line consists of exactly 64 characters except for the final line which will encode the remainder of the data (within the 64 character line boundary). Parsers MAY handle other line sizes. These requirements are consistent with PEM [RFC1421].

## 3. ABNF

The ABNF of the PKIX text encoding is:

```
pkixmsg    ::= preeb
```

```
               *eolWSP
               base64text
               posteb

preeb      ::= "-----BEGIN " label "-----" eol

posteb     ::= "-----END " label "-----" eol

base64char ::= ALPHA / DIGIT / "+" / "/"

base64pad  ::= "="

base64line ::= 1*base64char eol

base64finl ::= *base64char (base64pad eol base64pad /
               *2base64pad) eol
               ; ...AB= <CRLF> = <CRLF> is not good, but is valid

base64text ::= *base64line base64finl
        ; we could also use <encbinbody> from RFC 1421, which requires
        ; 16 groups of 4 chars, which means exactly 64 chars per
        ; line, except the final line, but this is more accurate

labelchar  ::= %x21-2C / %x2E-%7E   ; any printable character,
                                    ; except hyphen

label      ::= labelchar *(labelchar / labelchar "-" / SP) labelchar

eol        ::= CRLF / CR / LF

eolWSP     ::= WSP / CR / LF    ; compare with LWSP
```

**Figure 1: ABNF**

```
pkixmsgstrict    ::= preeb
                     strictbase64text
                     posteb

strictbase64finl ::= *15(4base64char) (4base64char / 3base64char
                     base64pad / 2base64char 2base64pad) eol

base64fullline   ::= 64base64char eol

strictbase64text ::= *base64fullline strictbase64finl
```

**Figure 2: ABNF (Strict)**

This specification RECOMMENDS that new implementations emit the strict format [abnf-strict-fig] specified above.

## 4. Guide

For convenience, this table summarizes the structures, encodings, and references in the following sections:

```
Sec. Label                  ASN.1 Type              Reference Module
----+----------------------+-----------------------+---------+----------
  5  CERTIFICATE            Certificate             [RFC5280] id-pkix1-e
  6  X.509 CRL              CertificateList         [RFC5280] id-pkix1-e
```

```
 7  CERTIFICATE REQUEST    CertificationRequest    [RFC2986] id-pkcs10
 8  PKCS7                  ContentInfo             [RFC2315] id-pkcs7*
 9  CMS                    ContentInfo             [RFC5652] id-cms2004
10  PRIVATE KEY            PrivateKeyInfo ::=      [RFC5208] id-pkcs8
                           OneAsymmetricKey        [RFC5958] id-aKPV1
11  ENCRYPTED PRIVATE KEY  EncryptedPrivateKeyInfo [RFC5958] id-aKPV1
12  ATTRIBUTE CERTIFICATE  AttributeCertificate    [RFC5755] id-acv2
13  PUBLIC KEY             SubjectPublicKeyInfo    [RFC5280] id-pkix1-e

ASN.1 Module Object Identifier Value Assignments
----------------------------------------------------------------------
id-pkixmod OBJECT IDENTIFIER ::= {iso(1) identified-organization(3)
           dod(6) internet(1) security(5) mechanisms(5) pkix(7) mod(0)}
id-pkix1-e OBJECT IDENTIFIER ::= {id-pkixmod pkix1-explicit(18)}
id-acv2    OBJECT IDENTIFIER ::= {id-pkixmod mod-attribute-cert-v2(61)}
id-pkcs    OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840)
                                  rsadsi(113549) pkcs(1)}
id-pkcs10  OBJECT IDENTIFIER ::= {id-pkcs 10 modules(1) pkcs-10(1)}
id-pkcs7   OBJECT IDENTIFIER ::= {id-pkcs 7 modules(0) pkcs-7(1)}
id-pkcs8   OBJECT IDENTIFIER ::= {id-pkcs 8 modules(1) pkcs-8(1)}
id-sm-mod  OBJECT IDENTIFIER ::= {id-pkcs 9 smime(16) modules(0)}
id-aKPV1   OBJECT IDENTIFIER ::= {id-sm-mod mod-asymmetricKeyPkgV1(50)}
id-cms2004 OBJECT IDENTIFIER ::= {id-sm-mod cms-2004(24)}
```

*This OID does not actually appear in PKCS #7 v1.5 [RFC2315]. It was defined in the ASN.1 module to PKCS #7 v1.6 [P7v1.6], and has been carried forward through PKCS #12 [RFC7292].

**Figure 3: Convenience Guide**

## 5. Text Encoding of Certificates

### 5.1. Encoding

Certificates are encoded using the `CERTIFICATE` label. The encoded data MUST be a BER (DER strongly preferred) encoded ASN.1 `Certificate` structure as described in section 4 of [RFC5280].

```
-----BEGIN CERTIFICATE-----
MIICLDCCAdKgAwIBAgIBADAKBggqhkjOPQQDAjB9MQswCQYDVQQGEwJCRTEPMA0G
A1UEChMGR251VExTMSUwIwYDVQQLExxHbnVUTFMgY2VydGlmaWNhdGUgYXV0aG9y
aXR5MQ8wDQYDVQQIEwZMZXV2ZW4xJTAjBgNVBAMTHEdudVRMUyBjZXJ0aWZpY2F0
ZSBhdXRob3JpdHkwHhcNMTEwNTIzMjAzODIxWhcNMTIxMjIyMDc0MTUxWjB9MQsw
CQYDVQQGEwJCRTEPMA0GA1UEChMGR251VExTMSUwIwYDVQQLExxHbnVUTFMgY2Vy
dGlmaWNhdGUgYXV0aG9yaXR5MQ8wDQYDVQQIEwZMZXV2ZW4xJTAjBgNVBAMTHEdu
dVRMUyBjZXJ0aWZpY2F0ZSBhdXRob3JpdHkwWTATBgcqhkjOPQIBBggqhkjOPQMB
BwNCAARS2I0jiuNn14Y2sSALCX3IybqiIJUvxUpj+oNfzngvj/Niyv2394BWnW4X
uQ4RTEiywK87WRcWMGgJB5kX/t2no0MwQTAPBgNVHRMBAf8EBTADAQH/MA8GA1Ud
DwEB/wQFAwMHBgAwHQYDVR0OBBYEFPC0gf6YEr+1KLlkQAPLzB9mTigDMAoGCCqG
SM49BAMCA0gAMEUCIDGuwD1KPyG+hRf88MeyMQcqOFZD0TbVleF+UsAGQ4enAiEA
l4wOuDwKQa+upc8GftXE2C//4mKANBC6It01gUaTIpo=
-----END CERTIFICATE-----
```

**Figure 4: Certificate Example**

Historically the label `X509 CERTIFICATE` and also less commonly `X.509 CERTIFICATE` have been used. Generators conforming to this document MUST generate `CERTIFICATE` labels and MUST NOT generate `X509 CERTIFICATE` or `X.509 CERTIFICATE` labels. Parsers are NOT RECOMMENDED to treat

`X509 CERTIFICATE` or `X.509 CERTIFICATE` as equivalent to `CERTIFICATE`, but a valid exception may be for backwards compatibility (potentially together with a warning).

## 5.2. Explanatory Text

Many tools are known to emit explanatory text before the BEGIN and after the END lines for PKIX certificates, more than any other type. If emitted, such text SHOULD be related to the certificate, such as providing a textual representation of key data elements in the certificate.

```
Subject: CN=Atlantis
Issuer: CN=Atlantis
Validity: from 7/9/2012 3:10:38 AM UTC to 7/9/2013 3:10:37 AM UTC
-----BEGIN CERTIFICATE-----
MIIBmTCCAUegAwIBAgIBKjAJBgUrDgMCHQUAMBMxETAPBgNVBAMTCEF0bGFudGlz
MB4XDTEyMDcwOTAzMTAzOFoXDTEzMDcwOTAzMTAzN1owEzERMA8GA1UEAxMIQXRs
YW50aXMwXDANBgkqhkiG9w0BAQEFAANLADBIAkEAu+BXo+miabDIHHx+yquqzqNh
Ryn/XtkJIIHVcYtHvIX+S1x5ErgMoHehycpoxbErZmVR4GCq1S2diNmRFZCRtQID
AQABo4GJMIGGMAwGA1UdEwEB/wQCMAAwIAYDVR0EAQH/BBYwFDAOMAwGCisGAQQB
gjcCARUDAgeAMB0GA1UdJQQWMBQGCCsGAQUFBwMCBggrBgEFBQcDAzA1BgNVHQEE
LjAsgBA0jOnSSuIHYmnVryHAdywMoRUwEzERMA8GA1UEAxMIQXRsYW50aXOCASow
CQYFKw4DAh0FAANBAKi6HRBaNEL5R0n56nvfclQNaXiDT174uf+lojzA4lhVInc0
ILwpnZ1izL4MlI9eCSHhVQBHEp2uQdXJB+d5Byg=
-----END CERTIFICATE-----
```

**Figure 5: Certificate Example with Explanatory Text**

## 5.3. File Extension

Although text encodings of PKIX structures can occur anywhere, many tools are known to offer an option to encode PKIX structures in this text encoding. To promote interoperability and to separate DER encodings from text encodings, This Internet-Draft RECOMMENDS that the extension `.crt` be used for this text encoding. Implementations should be aware that in spite of this recommendation, many tools still default to encode certificates in this text encoding with the extension `.cer`.

# 6. Text Encoding of Certificate Revocation Lists

Certificate Revocation Lists (CRLs) are encoded using the `X509 CRL` label. The encoded data MUST be a BER (DER strongly preferred) encoded ASN.1 `CertificateList` structure as described in Section 5 of [RFC5280].

```
-----BEGIN X509 CRL-----
MIIB9DCCAV8CAQEwCwYJKoZIhvcNAQEFMIIBCDEXMBUGA1UEChMOVmVyaVNpZ24s
IEluYy4xHzAdBgNVBAsTFlZlcmlTaWduIFRydXN0IE5ldHdvcmsxRjBEBgNVBAsT
PXd3dy52ZXJpc2lnbi5jb20vcmVwb3NpdG9yeS9SUEEgSW5jb3JwLiBieSBSZWYu
LExJQUIuTFREKGMpOTgxHjAcBgNVBAsTFVBlcnNvbmEgTm90IFZhbGlkYXRlZDEm
MCQGA1UECxMdRGlnaXRhbCBJRCBDbGFzcyAxIC0gTmV0c2NhcGUxGDAWBgNVBAMU
D1NpbW9uIEpvc2Vmc3NvbjEiMCAGCSqGSIb3DQEJARYTc2ltb25Aam9zZWZzc29u
Lm9yZxcNMDYxMjI3MDgwMjM0WhcNMDcwMjA3MDgwMjM1WjAjMCECEC4QNwPfRoWd
elUNpllhhTgXDTA2MTIyNzA4MDIzNFowCwYJKoZIhvcNAQEFA4GBAD0zX+J2hkcc
Nbrq1Dn5IKL8nXLgPGcHv1I/le1MNo9t1ohGQxB5HnFUkRPAY82fR6Epor4aHgVy
b+5y+neKN9Kn2mPF4iiun+a4o26CjJ0pArojCL1p8T0yyi9Xxvyc/ezaZ98HiIyP
c3DGMNR+oUmSjKZ0jIhAYmeLxaPHfQwR
-----END X509 CRL-----
```

**Figure 6: CRL Example**

Historically the label `CRL` has rarely been used. Today it is not common and many popular tools do not understand the label. Therefore, this document standardizes `X509 CRL` in order to promote interoperability and backwards-compatibility. Generators conforming to this document MUST generate `X509 CRL` labels and MUST NOT generate `CRL` labels. Parsers are NOT RECOMMENDED to treat `CRL` as equivalent to `X509 CRL`.

## 7. Text Encoding of PKCS #10 Certification Request Syntax

PKCS #10 Certification Requests are encoded using the `CERTIFICATE REQUEST` label. The encoded data MUST be a BER (DER strongly preferred) encoded ASN.1 `CertificationRequest` structure as described in [RFC2986].


```
-----BEGIN CERTIFICATE REQUEST-----
MIIBWDCCAQcCAQAwTjELMAkGA1UEBhMCU0UxJzAlBgNVBAoTHlNpbW9uIEpvc2Vm
c3NvbiBEYXRha29uc3VsdCBBQjEWMBQGA1UEAxMNam9zZWZzc29uLm9yZzBOMBAG
ByqGSM49AgEGBSuBBAAhAzoABLLPSkuXY0l66MbxVJ3Mot5FCFuqQfn6dTs+9/CM
EOlSwVej77tj56kj9R/j9Q+LfysX8FO9I5p3oGIwYAYJKoZIhvcNAQkOMVMwUTAY
BgNVHREEETAPgg1qb3NlZnNzb24ub3JnMAwGA1UdEwEB/wQCMAAwDwYDVR0PAQH/
BAUDAwegADAWBgNVHSUBAf8EDDAKBggrBgEFBQcDATAKBggqhkjOPQQDAgM/ADA8
AhxBvfhxPFfbBbsE1NoFmCUczOFApEuQVUw3ZP69AhwWXk3dgSUsKnuwL5g/ftAY
dEQc8B8jAcnuOrfU
-----END CERTIFICATE REQUEST-----
```


**Figure 7: PKCS #10 Example**

The label `NEW CERTIFICATE REQUEST` is also in wide use. Generators conforming to this document MUST generate `CERTIFICATE REQUEST` labels. Parsers MAY treat `NEW CERTIFICATE REQUEST` as equivalent to `CERTIFICATE REQUEST`.

## 8. Text Encoding of PKCS #7 Cryptographic Message Syntax

PKCS #7 Cryptographic Message Syntax structures are encoded using the `PKCS7` label. The encoded data MUST be a BER encoded ASN.1 `ContentInfo` structure as described in [RFC2315].


```
-----BEGIN PKCS7-----
MIHjBgsqhkiG9w0BCRABF6CB0zCB0AIBADFho18CAQCgGwYJKoZIhvcNAQUMMA4E
CLfrI6dr0gUWAgITiDAjBgsqhkiG9w0BCRADCTAUBggqhkiG9w0DBwQIZpECRWtz
u5kEGDCjerXY8odQ7EEEromZJvAurk/j81IrozBSBgkqhkiG9w0BBwEwMwYLKoZI
hvcNAQkQAw8wJDAUBggqhkiG9w0DBwQI0tCBcU09nxEwDAYIKwYBBQUIAQIFAIAQ
OsYGYUFdAH0RNc1p4VbKEAQUM2Xo8PMHBoYdqEcsbTodlCFAZH4=
-----END PKCS7-----
```


**Figure 8: PKCS #7 Example**

The label `CERTIFICATE CHAIN` has been in use to denote a degenerative PKCS #7 structure that contains only a list of certificates. Several modern tools do not support this label. Generators MUST NOT generate the `CERTIFICATE CHAIN` label. Parsers are NOT RECOMMENDED to treat `CERTIFICATE CHAIN` as equivalent to `PKCS7`.

PKCS #7 is an old standard that has long been superseded by CMS [RFC5652]. Implementations SHOULD NOT generate PKCS #7 when CMS is an alternative.

## 9. Text Encoding of Cryptographic Message Syntax

Cryptographic Message Syntax structures are encoded using the `CMS` label. The encoded data MUST be a BER encoded ASN.1 `ContentInfo` structure as described in [RFC5652].

```
-----BEGIN CMS-----
```

```
MIGDBgsqhkiG9w0BCRABCaB0MHICAQAwDQYLKoZIhvcNAQkQAwgwXgYJKoZIhvcN
AQcBoFEET3icc87PK0nNK9ENqSxItVIoSa0o0S/ISczMs1ZIzkgsKk4tsQ0N1nUM
dvb05OXi5XLPLEtViMwvLVLwSE0sKlFIVHAqSk3MBkkBAJv0Fx0=
-----END CMS-----
```


**Figure 9: CMS Example**

CMS is the IETF successor to PKCS #7. Section 1.1.1 of [RFC5652] describes the changes since PKCS #7 v1.5. Implementations SHOULD generate CMS when it is an alternative, promoting interoperability and forwards-compatibility.

## 10. Text Encoding of PKCS #8 Private Key Info, and One Asymmetric Key

Unencrypted PKCS #8 Private Key Information Syntax structures (`PrivateKeyInfo`), renamed to Asymmetric Key Packages (`OneAsymmetricKey`), are encoded using the `PRIVATE KEY` label. The encoded data MUST be a BER (DER preferred) encoded ASN.1 `PrivateKeyInfo` structure as described in PKCS #8 [RFC5208], or a `OneAsymmetricKey` structure as described in [RFC5958]. The two are semantically identical, and can be distinguished by version number.


```
-----BEGIN PRIVATE KEY-----
MIGEAgEAMBAGByqGSM49AgEGBSuBBAAKBG0wawIBAQQgVcB/UNPxalR9zDYAjQIf
jojUDiQuGnSJrFEEzZPT/92hRANCAASc7UJtgnF/abqWM60T3XNJEzBv5ez9TdwK
H0M6xpM2q+53wmsN/eYLdgtjgBd3DBmHtPilCkiFICXyaA8z9LkJ
-----END PRIVATE KEY-----
```


**Figure 10: PKCS #8 PrivateKeyInfo Example**

## 11. Text Encoding of PKCS #8 Encrypted Private Key Info

Encrypted PKCS #8 Private Key Information Syntax structures (`EncryptedPrivateKeyInfo`), called the same in [RFC5958], are encoded using the `ENCRYPTED PRIVATE KEY` label. The encoded data MUST be a BER (DER preferred) encoded ASN.1 `EncryptedPrivateKeyInfo` structure as described in PKCS #8 [RFC5208] and [RFC5958].


```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIHNMEAGCSqGSIb3DQEFDTAzMBsGCSqGSIb3DQEFDDAOBAghhICA6T/51QICCAAw
FAYIKoZIhvcNAwcECBCxDgvI59i9BIGIY3CAqlMNBgaSI5QiiWVNJ3IpfLnEiEsW
Z0JIoHyRmKK/+cr9QPLnzxImm0TR9s4JrG3CilzTWvb0jIvbG3hu0zyFPraoMkap
8eRzWsIvC5SVel+CSjoS2mVS87cyjlD+txrmrXOVYDE+eTgMLbrLmsWh3QkCTRtF
QC7k0NNzUHTV9yGDwfqMbw==
-----END ENCRYPTED PRIVATE KEY-----
```


**Figure 11: PKCS #8 EncryptedPrivateKeyInfo Example**

## 12. Text Encoding of Attribute Certificates

Attribute certificates are encoded using the `ATTRIBUTE CERTIFICATE` label. The encoded data MUST be a BER (DER strongly preferred) encoded ASN.1 `AttributeCertificate` structure as described in [RFC5755].


```
-----BEGIN ATTRIBUTE CERTIFICATE-----
MIICKzCCAZQCAQEwgZeggZQwgYmkgYYwgYMxCzAJBgNVBAYTAlVTMREwDwYDVQQI
DAhOZXcgWW9yazEUMBIGA1UEBwwLU3RvbnkgQnJvb2sxDzANBgNVBAoMBkNTRTU
MjE6MDgGA1UEAwwxU2NvdHQgU3RhbGxlci9lbWFpbEFkZHJlc3M9c3N0YWxsZXJA
aWMuc3VueXNiLmVkdQIGARWrgUUSoIGMMIGJpIGGMIGDMQswCQYDVQQGEwJVUzER
MA8GA1UECAwITmV3IFlvcmsxFDASBgNVBAcMC1N0b255IEJyb29rMQ8wDQYDVQQK
DAZDU0U1OTIxOjA4BgNVBAMMMVNjb3R0IFN0YWxsZXIvZW1haWxBZGRyZXNzPXNz
```

```
dGFsbGVyQGljLnN1bnlzYi5lZHUwDQYJKoZIhvcNAQEFBQACBgEVq4FFSjAiGA8z
OTA3MDIwMTA1MDAwMFoYDzM5MTEwMTMxMDUwMDAwWjArMCkGA1UYSDEiMCCGHmh0
dHA6Ly9pZGVyYXNobi5vcmcvaW5kZXguaHRtbDANBgkqhkiG9w0BAQUFAAOBgQAV
M9axFPXXozEFcer06bj9MCBBCQLtAM7ZXcZjcxyva7xCBDmtZXPYUluHf5OcWPJz
5XPus/xS9wBgtlM3fldIKNyNO8RsMp6Ocx+PGlICc7zpZiGmCYLl64lAEGPO/bsw
Smluak1aZIttePeTAHeJJs8izNJ5aR3Wcd3A5gLztQ==
-----END ATTRIBUTE CERTIFICATE-----
```


**Figure 12: Attribute Certificate Example**

## 13. Text Encoding of Subject Public Key Info

Public keys are encoded using the `PUBLIC KEY` label. The encoded data MUST be a BER (DER preferred) encoded ASN.1 `SubjectPublicKeyInfo` structure as described in Section 4.1.2.7 of [RFC5280].


```
-----BEGIN PUBLIC KEY-----
MHYwEAYHKoZIzj0CAQYFK4EEACIDYgAEn1LlwLN/KBYQRVH6HfIMTzfEqJOVztLe
kLchp2hi78cCaMY81FBlYs8J9l7krc+M4aBeCGYFjba+hiXttJWPL7ydlE+5UG4U
Nkn3Eos8EiZByi9DVsyfy9eejh+8AXgp
-----END PUBLIC KEY-----
```


**Figure 13: Subject Public Key Info Example**

## 14. Security Considerations

Data in this format often originates from untrusted sources, thus parsers must be prepared to handle unexpected data without causing security vulnerabilities.

Implementers building implementations that rely on canonical representation or the ability to fingerprint a particular data object need to understand that this Internet-Draft does not define canonical encodings. The first ambiguity is introduced by permitting the text-encoded representation instead of the binary BER or DER encodings, but further ambiguities arise when multiple labels are treated as similar. Variations of whitespace and non-base64 alphabetic characters can create further ambiguities. Data encoding ambiguities also create opportunities for side channels. If canonical encodings are desired, the encoded structure must be decoded and processed into a canonical form (namely, DER encoding).

## 15. IANA Considerations

This document implies no IANA Considerations.

## 16. Acknowledgements

Peter Gutmann suggested to document labels for Attribute Certificates and PKCS #7 messages, and to add examples for the non-standard variants. Dr. Stephen Henson suggested distinguishing when BER versus DER are appropriate or necessary.

## 17. References

### 17.1. Normative References


**[RFC2119]** Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

**[RFC2315]** Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, March 1998.

**[RFC2986]** Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, November 2000.

**[RFC4648]** Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

**[RFC5208]** Kaliski, B., "Public-Key Cryptography Standards (PKCS) #8: Private-Key Information Syntax Specification Version 1.2", RFC 5208, May 2008.

**[RFC5280]** Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R. and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

**[RFC5652]** Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

**[RFC5755]** Farrell, S., Housley, R. and S. Turner, "An Internet Attribute Certificate Profile for Authorization", RFC 5755, January 2010.

**[RFC5958]** Turner, S., "Asymmetric Key Packages", RFC 5958, August 2010.

**[X.690]** International Telecommunications Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1:2008, November 2008.

## 17.2. Informative References

**[RFC0934]** Rose, M. and E. Stefferud, "Proposed standard for message encapsulation", RFC 934, January 1985.

**[RFC1421]** Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures", RFC 1421, February 1993.

**[RFC2015]** Elkins, M., "MIME Security with Pretty Good Privacy (PGP)", RFC 2015, October 1996.

**[RFC4716]** Galbraith, J. and R. Thayer, "The Secure Shell (SSH) Public Key File Format", RFC 4716, November 2006.

**[RFC7292]** Moriarty, K., Nystrom, M., Parkinson, S., Rusch, A. and M. Scott, "PKCS #12: Personal Information Exchange Syntax v1.1", RFC 7292, July 2014.

**[P7v1.6]** Kaliski, B. and K. Kingdon, "Extensions and Revisions to PKCS #7 (Version 1.6 Bulletin)", May 1997.

**[X.509SG]** Gutmann, P., "X.509 Style Guide", October 2000.

## Appendix A. Non-Conforming Examples

This section contains examples for the non-recommended label variants described earlier in this document. As discussed earlier, supporting these are not required and sometimes discouraged. Still, they can be useful for interoperability testing and for easy reference.


```
-----BEGIN X509 CERTIFICATE-----
MIIBHDCBxaADAgECAgIcxzAJBgcqhkjOPQQBMBAxDjAMBgNVBAMUBVBLSVghMB4X
DTE0MDkxNDA2MTU1MFoXDTI0MDkxNDA2MTU1MFowEDEOMAwGA1UEAxQFUEtJWCEw
WTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAATwoQSr863QrR0PoRIYQ96H7WykDePH
Wa0eVAE24bth43wCNc+U5aZ761dhGhSSJkVWRgVH5+prLIr+nzfIq+X4oxAwDjAM
BgNVHRMBAf8EAjAAMAkGByqGSM49BAEDRwAwRAIfMdKS5F63lMnWVhi7uaKJzKCs
NnY/OKgBex6MIEAv2AIhAI2GdvfL+mGvhyPZE+JxRxWChmggb5/9eHdUcmW/jkOH
-----END X509 CERTIFICATE-----
```


**Figure 14: Non-standard 'X509' Certificate Example**


```
-----BEGIN X.509 CERTIFICATE-----
MIIBHDCBxaADAgECAgIcxzAJBgcqhkjOPQQBMBAxDjAMBgNVBAMUBVBLSVghMB4X
DTE0MDkxNDA2MTU1MFoXDTI0MDkxNDA2MTU1MFowEDEOMAwGA1UEAxQFUEtJWCEw
WTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAATwoQSr863QrR0PoRIYQ96H7WykDePH
Wa0eVAE24bth43wCNc+U5aZ761dhGhSSJkVWRgVH5+prLIr+nzfIq+X4oxAwDjAM
BgNVHRMBAf8EAjAAMAkGByqGSM49BAEDRwAwRAIfMdKS5F63lMnWVhi7uaKJzKCs
NnY/OKgBex6MIEAv2AIhAI2GdvfL+mGvhyPZE+JxRxWChmggb5/9eHdUcmW/jkOH
-----END X.509 CERTIFICATE-----
```


**Figure 15: Non-standard 'X.509' Certificate Example**

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBWDCCAQcCAQAwTjELMAkGA1UEBhMCU0UxJzAlBgNVBAoTHlNpbW9uIEpvc2Vm
c3NvbiBEYXRha29uc3VsdCBBQjEWMBQGA1UEAxMNam9zZWZzc29uLm9yZzBOMBAG
ByqGSM49AgEGBSuBBAAhAzoABLLPSkuXY0l66MbxVJ3Mot5FCFuqQfn6dTs+9/CM
EOlSwVej77tj56kj9R/j9Q+LfysX8FO9I5p3oGIwYAYJKoZIhvcNAQkOMVMwUTAY
BgNVHREEETAPgg1qb3NlZnNzb24ub3JnMAwGA1UdEwEB/wQCMAAwDwYDVR0PAQH/
BAUDAwegADAWBgNVHSUBAf8EDDAKBggrBgEFBQcDATAKBggqhkjOPQQDAgM/ADA8
AhxBvfhxPFfbBbsE1NoFmCUczOFApEuQVUw3ZP69AhwWXk3dgSUsKnuwL5g/ftAY
dEQc8B8jAcnuOrfU
-----END NEW CERTIFICATE REQUEST-----
```


**Figure 16: Non-standard 'NEW' PKCS #10 Example**


```
-----BEGIN CERTIFICATE CHAIN-----
MIHjBgsqhkiG9w0BCRABF6CB0zCB0AIBADFho18CAQCgGwYJKoZIhvcNAQUMMA4E
CLfrI6dr0gUWAgITiDAjBgsqhkiG9w0BCRADCTAUBggqhkiG9w0DBwQIZpECRWtz
u5kEGDCjerXY8odQ7EEEromZJvAurk/j81IrozBSBgkqhkiG9w0BBwEwMwYLKoZI
hvcNAQkQAw8wJDAUBggqhkiG9w0DBwQI0tCBcU09nxEwDAYIKwYBBQUIAQIFAIAQ
OsYGYUFdAH0RNc1p4VbKEAQUM2Xo8PMHBoYdqEcsbTodlCFAZH4=
-----END CERTIFICATE CHAIN-----
```


**Figure 17: Non-standard 'CERTIFICATE CHAIN' Example**

## Appendix B. DER Expectations

This appendix is informative. Consult the respective standards for the normative rules.

DER is a restricted profile of BER [X.690]; thus all DER encodings of data values are BER encodings, but just one of the BER encodings is the DER encoding for a data value. Canonical encoding matters when performing cryptographic operations; additionally, canonical encoding has certain efficiency advantages for parsers. There are three principal reasons to do encode with DER:

1. A digital signature is (supposed to be) computed over the DER encoding of the semantic content, so providing anything other than the DER encoding is senseless. (In practice, an implementer might choose to have an implementation parse and digest the data as-is, but this practice amounts to guesswork.)
2. In practice, cryptographic hashes are computed over the DER encoding for identification.
3. In practice, the content is small. DER always encodes data values in definite length form (where the length is stated at the beginning of the encoding); thus, a parser can anticipate memory or resource usage up-front.

```
Sec. Label                  Reasons
----+----------------------+-------
  5  CERTIFICATE            1  2 ~3
  6  X.509 CRL              1
  7  CERTIFICATE REQUEST    1    ~3
  8  PKCS7                  *
  9  CMS                    *
 10  PRIVATE KEY                  3
 11  ENCRYPTED PRIVATE KEY        3
 12  ATTRIBUTE CERTIFICATE  1    ~3
 13  PUBLIC KEY                2  3
```

Figure 18 matches the structures in this document with the particular reasons for DER encoding:

*Cryptographic Message Syntax is designed for content of any length; indefinite length encoding enables one-pass processing (streaming) when generating the encoding. Only certain parts, namely signed and authenticated attributes,

need to be DER encoded.
~Although not always "small", these encoded structures should not be particularly "large" (e.g., more than 16 kilobytes). The parser ought to be informed of large things up-front in any event, which is yet another reason to DER encode these things in the first place.

**Figure 18: Guide for DER Encoding**

## Authors' Addresses

**Simon Josefsson**
SJD AB
Johan Olof Wallins Väg 13
Solna, 171 64
SE
EMail: simon@josefsson.org
URI: http://josefsson.org/

**Sean Leonard**
Penango, Inc.
5900 Wilshire Boulevard
21st Floor
Los Angeles, CA 90036
USA
EMail: dev+ietf@seantek.com
URI: http://www.penango.com/