Text Encodings of PKIX and CMS Structures
draft-josefsson-pkix-textual-05

Abstract

This document describes and discusses the text encodings of Public-Key Infrastructure using X.509 (PKIX) Certificates, PKIX Certificate Revocation Lists (CRLs), PKCS #10 Certification Request Syntax, PKCS #7 structures, Cryptographic Message Syntax (CMS), PKCS #8 Private-Key Information Syntax, and Attribute Certificates. 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.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 4, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

Several security-related standards used on the Internet define data formats that are normally encoded using Distinguished Encoding Rules (DER) [CCITT.X690.2002], which is a binary data format. This document is about text encodings of some of these formats:

1. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile [RFC5280], for both Certificates and Certificate Revocation Lists (CRLs).
2. PKCS #10: Certification Request Syntax [RFC2986].
3. PKCS #7: Cryptographic Message Syntax [RFC2315].
5. PKCS #8: Private-Key Information Syntax [RFC5208] and One Asymmetric Key (in Asymmetric Key Package [RFC5958]).

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 [X509SG] 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.

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.
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
4. Text Encoding of PKIX Certificates

4.1. Encoding

PKIX certificates are encoded using the **CERTIFICATE** label. The encoded data MUST be a DER encoded ASN.1 Certificate structure as described in section 4 of [RFC5280].

```
-----BEGIN CERTIFICATE-----
MIICLDCCAdKgAwIBAgIBADAKBggqhkjOPQQDAjB9MQswCQYDVQQGEwJCRTEPMA0G
A1UEChMGR251VExTMSUwIwYDVQQLEExhMjMwNTAzMjEwMTAzMjEwMHo8MA0GA1UE
CgQGM28wGAYDVQQKExJXaW50ZW5pZ2h0MTAwMDAeFw0yMDEwMTQzMDEwMDAwOzAf
BgNVBAsTC1N0YW5jZWN0b3J5MRQGANEwHhcNMTEwNzAzMDEwMDEwMTAwWjBBgmart
aT5MGA1UdDgQMAgEBDQIBBAIBCwUDBgNVHRMBAf8EBTADAQH/MA8GA1UdDwEB/wQF
AwMEAgECMAoGCSqGSIb3DQEJEQBaAHqWl2hIHRoZXRhLmNvbTAMBgHAwhEAUGFy
единение—
-----END CERTIFICATE-----
```

**Figure 3: Certificate Example**

Historically the label **X509 CERTIFICATE** and also, less common, **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).

4.2. 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.

**Figure 4: Certificate Example with Explanatory Text**

### 4.3. 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".

### 5. Text Encoding of PKIX CRLs

PKIX CRLs are encoded using the **X509 CRL** label. The encoded data MUST be a DER encoded **CertificateList** structure as described in Section 5 of [RFC5280].

**Figure 5: 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
6. 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 DER encoded ASN.1 CertificationRequest structure as described in [RFC2986].

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBWDCCAQcCAQAwTjELMAkGA1UEBhMCVVMxLjAgMB8GA1UdDwEB/woYDวยWhf3EICwIB
-----END CERTIFICATE REQUEST-----
```

Figure 6: 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.

7. 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 DER encoded ASN.1 ContentInfo structure as described in [RFC2315].

```
-----BEGIN PKCS7-----
MIHjBgsqhkiG9w0BCRBAP6CB0zCB0AIBADFho18CAQCCgGwYJKoZIhvcNAQUMMA4E
CLfrI6dr0gUWAqTiDAjBGskhikG9w0BCRBADCTAUBgghhkiG9w0DBwQ1p5ECRtz
u5kEGDCjerXX8odQ7EEeromZJueAvrkJ81IrozBBSkgkhkiG9w0BBwWvYlKoZI
hvcNAQkQAw8wJDAUBgghhkiG9w0DBwQ10tCBoU09nxEwDAYikYBBQUIAQ1FAIAQ
OsYGyuFha0RNC1p4VbKEAQuM2o8PMHBoYdqEcsbTodlCFAZH4=
-----END PKCS7-----
```

Figure 7: 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.

8. Text Encoding of Cryptographic Message Syntax
Cryptographic Message Syntax structures are encoded using the CMS label. The encoded data MUST be a DER encoded ASN.1 ContentInfo structure as described in [RFC5652].

-----BEGIN CMS-----
MIGDgsgqhkiC9w0BCRABCAb0MICAQAwDQYLKoZIhvcNAQkQAwgwYJKoZIhvcN
AQcBoFEETTicc87PK0oNKN9ENq0xITVoSa0o0S/TSczM12IzkgsKk4tsQ0N1nUM
dVb05OXi5XLPLEtViMVwVLWwSE0sKlFIVHAqSk3MBkkBAJv0Fx0=
-----END CMS-----

Figure 8: 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.

9. 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 DER 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-----
MIGEAgEAMBAGByqGSM49AgEGBSuBBAAKBG0wawIBAgIBAQQgVcB/UNPxa1R9zDYAjQIf
joJUDiQnSJsFEEzZPT/92hRANCAASc7UJtgnF/abqWN60T3XNJEzBv5ez9TdwK
H0M6xp5g+53wmsN/eYLdqtjg8d3DMhTpiLckiFICXyaA8z9LkJ
-----END PRIVATE KEY-----

Figure 9: PKCS #8 PrivateKeyInfo Example

10. 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 DER encoded ASN.1 EncryptedPrivateKeyInfo structure as described in PKCS #8 [RFC5208] and [RFC5958].

-----BEGIN ENCRYPTED PRIVATE KEY-----
MTINNEAGCSqGSIb3DQEFDTAzMBsGCsGSIb3DQEFDADA0BAgghICA6T/5lQICCAAw
FAYIKoZIhvcNAwCCEBCxQv519i9B1GYY3CAq1MBgaSI5qiiWNJ3IpfL1iEsW
Z0JiOHyRmKK/+cr9QPlnZIimOTR9s4J5G3c1izTivb0j1vbg3h50zyFPraoMkap
8eRzwIsVc5SVel+CSjoS32MSV87cyjL+txxmrXOYD+eTgMBrLmsWh3QkCٹRtF
QC7k0NNzUHTV9yGDwfgMb==
-----END ENCRYPTED PRIVATE KEY-----

Figure 10: PKCS #8 EncryptedPrivateKeyInfo Example
11. Text Encoding of Attribute Certificates

Attribute certificates are encoded using the *ATTRIBUTE CERTIFICATE* label. The encoded data MUST be a DER encoded ASN.1 AttributeCertificate structure as described in [RFC5755].

```
-----BEGIN ATTRIBUTE CERTIFICATE-----
MIICKzCCAZQCAQEwgZeggZQwgYmkgYYwgYMxCzAJBqNVBAyTA1vTMREwDwYDVQQI
DA02XcgW9yazEUMBIGA1UEBwwLUE3rvbnkgQnJvb2sxDzANBgNVBAoMBkNTRTU5
MjE6MDgGA1UEAwwxU2NvdHQgU3RhbGxlci9lci9lbWFpbEFkZHJlc3M9c3N0YWxsZXJA
aW MMC3VyeXNlLmVkdQIGARWrgUUSoIGMIGjIgMIGhenh0dHA6Ly9pZGVyYXNobi5vcmcvaW5kZXguaHRtbDANBgkqhkiG9w0BAQUFAAOBgQAV
M9axFExXzzEFcero6bju9MCBCQmLAtM7ZXc2jcxxyva7xCBDmzXFPSULBuHf50cWJz
5XPus/xS9wBgt1M3f1D1KkNyN08RsMmp6Ocx+PGLlCC7zp2iGmC1l641AGPO/bsw
SmLuaklaZlttePeTAHeJJS8izNJ5aR3WcdA5gLztQ==
-----END ATTRIBUTE CERTIFICATE-----
```

**Figure 11: Attribute Certificate Example**

12. Security Considerations

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

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

13. IANA Considerations

This document implies no IANA Considerations.

14. Acknowledgements

Peter Gutmann suggested to document labels for Attribute Certificates and PKCS #7 messages, and to add examples for the non-standard variants.

15. References
15.1. Normative References


15.2. Informative References


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-----
MIICLDCCAdKgAwIBAgIBADAKBggqhkjOPQDAjB9MQswCQYDVQQGEwJCVQREjABMA0G
A1UEChMGR251VExTMSUwIwYDVQQIEwZEMZV2W44xMzBjBgNVBAoTVEF0dXNlMjAx
NDAwCFczMDAwMDAwMC0wIzEpMA0GCSqGSIb3DQEBCwUAAnlJ7x+c8KuJ5L6Mf7/3K
7FjFbf438sLD2T/WsJFjOzv+yHv7oa8G8kqSGmmoJq5c+i7ZAFs5iV2dZd1+J+6/M
-----END X509 CERTIFICATE-----

Figure 12: Non-standard 'X509' Certificate Example

-----BEGIN X.509 CERTIFICATE-----
MIICLDCCAdKgAwIBAgIBADAKBggqhkjOPQDAjB9MQswCQYDVQQGEwJCVQREjABMA0G
A1UEChMGR251VExTMSUwIwYDVQQIEwZEMZV2W44xMzBjBgNVBAoTVEF0dXNlMjAx
NDAwCFczMDAwMDAwMC0wIzEpMA0GCSqGSIb3DQEBCwUAAnlJ7x+c8KuJ5L6Mf7/3K
7FjFbf438sLD2T/WsJFjOzv+yHv7oa8G8kqSGmmoJq5c+i7ZAFs5iV2dZd1+J+6/M
-----END X.509 CERTIFICATE-----

Figure 13: Non-standard 'X.509' Certificate Example

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBWDCCAQcCAQAwTjELMAkGA1UEBhMCU0UxJzAlBgNVBAoTHlNpbW9uIEpvc2Vz
aXR5MQo8wDQYDVQQIEwZEMZV2W44xMzBjBgNVBAoTVEF0dXNlMjAxNDAwCFczMDAw
MDAwMC0wIzEpMA0GCSqGSIb3DQEBCwUAAnlJ7x+c8KuJ5L6Mf7/3K7FjFbf438sLD2
T/WsJFjOzv+yHv7oa8G8kqSGmmoJq5c+i7ZAFs5iV2dZd1+J+6/M
-----END NEW CERTIFICATE REQUEST-----

Figure 14: Non-standard 'NEW' PKCS #10 Example

-----BEGIN CERTIFICATE CHAIN-----
MIIBqSBgA0GMEUClDGVd1KPYG+hRf88MeyMqcgOFZD0TbVl+eF+UsAQ4enAiEA
l4wOuDwKqa+upc8GftX2E2C//4mKANBC6It0lgUaTIp=
-----END X.509 CERTIFICATE-----

Figure 15: Non-standard X.509 certificate example.

----BEGIN CERTIFICATE REQUEST-----
MIIBqSBgA0GMEUClDGVd1KPYG+hRf88MeyMqcgOFZD0TbVl+eF+UsAQ4enAiEA
l4wOuDwKqa+upc8GftX2E2C//4mKANBC6It0lgUaTIp=
-----END CERTIFICATE REQUEST-----

Figure 16: Non-standard certificate request example.
Figure 15: Non-standard 'CERTIFICATE CHAIN' Example

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/