Internet Engineering Task Force (IETF)
Request for Comments: 5958
Obsoletes: 5208
Category: Standards Track
ISSN: 2070-1721

S. Turner
IECA

August 2010

Asymmetric Key Packages

Abstract

This document defines the syntax for private-key information and a content type for it. Private-key information includes a private key for a specified public-key algorithm and a set of attributes. The Cryptographic Message Syntax (CMS), as defined in RFC 5652, can be used to digitally sign, digest, authenticate, or encrypt the asymmetric key format content type. This document obsoletes RFC 5208.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5958.

1. Introduction

This document defines the syntax for private-key information and a Cryptographic Message Syntax (CMS) [RFC5652] content type for it. Private-key information includes a private key for a specified public-key algorithm and a set of attributes. The CMS can be used to digitally sign, digest, authenticate, or encrypt the asymmetric key format content type. This document obsoletes PKCS #8 v1.2 [RFC5208].

1.1. Requirements Terminology

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

1.2. ASN.1 Syntax Notation

The key package is defined using ASN.1 [X.680], [X.681], [X.682], and [X.683].

1.3. Summary of Updates to RFC 5208

The following summarizes the updates to [RFC5208]:

- Changed the name "PrivateKeyInfo" to "OneAsymmetricKey". This reflects the addition of the publicKey field to allow both parts of the asymmetric key to be conveyed separately. Not all algorithms will use both fields; however, the publicKey field was added for completeness.

- Defined Asymmetric Key Package CMS content type.

- Removed redundant IMPLICIT from attributes.

- Added publicKey to OneAsymmetricKey and updated the version number.

- Added that PKCS #9 attributes may be supported.

- Added discussion of compatibility with other private-key formats.

- Added requirements for encoding rule set.

- Changed imports from PKCS #5 to [RFC5912] and [RFC5911].

- Replaced ALGORITHM-IDENTIFIER with ALGORITHM from [RFC5912].

- Registers application/pkcs8 media type and .p8 file extension.

2. Asymmetric Key Package CMS Content Type

The asymmetric key package CMS content type is used to transfer one or more plaintext asymmetric keys from one party to another. An asymmetric key package MAY be encapsulated in one or more CMS protecting content types (see Section 4). Earlier versions of this specification [RFC5208] did not specify a particular encoding rule set, but generators SHOULD use DER [X.690] and receivers MUST support BER [X.690], which also includes DER [X.690].

The asymmetric key package content type has the following syntax:

```
ct-asymmetric-key-package CONTENT-TYPE ::=
  { AsymmetricKeyPackage IDENTIFIED BY id-ct-KP-aKeyPackage }

id-ct-KP-aKeyPackage OBJECT IDENTIFIER ::=
  { joint-iso-itu-t(2) country(16) us(840) organization(1)
    gov(101) dod(2) infosec(1) formats(2)
    key-package-content-types(78) 5
  }

AsymmetricKeyPackage ::= SEQUENCE SIZE (1..MAX) OF OneAsymmetricKey

OneAsymmetricKey ::= SEQUENCE {
  version                   Version,
  privateKeyAlgorithm       PrivateKeyAlgorithmIdentifier,
  privateKey                PrivateKey,
  attributes            [0] Attributes OPTIONAL,
  ...,
  [[2: publicKey        [1] PublicKey OPTIONAL ]],
  ...
}
```

```
PrivateKeyInfo ::= OneAsymmetricKey

-- PrivateKeyInfo is used by [P12].  If any items tagged as version
-- 2 are used, the version must be v2, else the version should be
-- v1.  When v1, PrivateKeyInfo is the same as it was in [RFC5208].

Version ::= INTEGER { v1(0), v2(1) } (v1, ..., v2)

PrivateKeyAlgorithmIdentifier ::= AlgorithmIdentifier
                                    { PUBLIC-KEY,
                                      { PrivateKeyAlgorithms } }

PrivateKey ::= OCTET STRING
                    -- Content varies based on type of key.  The
                    -- algorithm identifier dictates the format of
                    -- the key.

PublicKey ::= BIT STRING
                    -- Content varies based on type of key.  The
                    -- algorithm identifier dictates the format of
                    -- the key.

Attributes ::= SET OF Attribute { { OneAsymmetricKeyAttributes } }
```

The AsymmetricKeyPackage contains one or more OneAsymmetricKey elements.

The syntax of OneAsymmetricKey accommodates a version number, an indication of the asymmetric algorithm to be used with the private key, a private key, optional keying material attributes (e.g., userCertificate from [X.520]), and an optional public key. In general, either the public key or the certificate will be present. In very rare cases will both the public key and the certificate be present as this includes two copies of the public key. OneAsymmetricKey renames the PrivateKeyInfo syntax defined in [RFC5208]. The new name better reflects the ability to carry both private- and public-key components. Backwards compatibility with the original PrivateKeyInfo is preserved via version number. The fields in OneAsymmetricKey are used as follows:

- version identifies the version of OneAsymmetricKey. If publicKey is present, then version is set to v2 else version is set to v1.

- privateKeyAlgorithm identifies the private-key algorithm and optionally contains parameters associated with the asymmetric key pair. The algorithm is identified by an object identifier (OID) and the format of the parameters depends on the OID, but the PrivateKeyAlgorithms information object set restricts the

permissible OIDs. The value placed in privateKeyAlgorithmIdentifier is the value an originator would apply to indicate which algorithm is to be used with the private key.

- privateKey is an OCTET STRING that contains the value of the private key. The interpretation of the content is defined in the registration of the private-key algorithm. For example, a DSA key is an INTEGER, an RSA key is represented as RSAPrivateKey as defined in [RFC3447], and an Elliptic Curve Cryptography (ECC) key is represented as ECPrivateKey as defined in [RFC5915].

- attributes is OPTIONAL. It contains information corresponding to the public key (e.g., certificates). The attributes field uses the class ATTRIBUTE which is restricted by the OneAsymmetricKeyAttributes information object set. OneAsymmetricKeyAttributes is an open ended set in this document. Others documents can constrain these values. Attributes from [RFC2985] MAY be supported.

- publicKey is OPTIONAL. When present, it contains the public key encoded in a BIT STRING. The structure within the BIT STRING, if any, depends on the privateKeyAlgorithm. For example, a DSA key is an INTEGER. Note that RSA public keys are included in RSAPrivateKey (i.e., n and e are present), as per [RFC3447], and ECC public keys are included in ECPrivateKey (i.e., in the publicKey field), as per [RFC5915].

3. Encrypted Private Key Info

This section gives the syntax for encrypted private-key information, which is used by [P12].

Encrypted private-key information shall have ASN.1 type EncryptedPrivateKeyInfo:

```
EncryptedPrivateKeyInfo ::= SEQUENCE {
  encryptionAlgorithm  EncryptionAlgorithmIdentifier,
  encryptedData        EncryptedData }

EncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
                                    { CONTENT-ENCRYPTION,
                                      { KeyEncryptionAlgorithms } }

EncryptedData ::= OCTET STRING
```

The fields in EncryptedPrivateKeyInfo are used as follows:

- encryptionAlgorithm identifies the algorithm under which the private-key information is encrypted.

- encryptedData is the result of encrypting the private-key information (i.e., the PrivateKeyInfo).

The encryption process involves the following two steps:

1. The private-key information is encoded, yielding an octet string. Generators SHOULD use DER [X.690] and receivers MUST support BER [X.690], which also includes DER [X.690].

2. The result of step 1 is encrypted with the secret key to give an octet string, the result of the encryption process.

4. Protecting the AsymmetricKeyPackage

CMS protecting content types, [RFC5652] and [RFC5083], can be used to provide security to the AsymmetricKeyPackage:

- SignedData can be used to apply a digital signature to the AsymmetricKeyPackage.

- EncryptedData can be used to encrypt the AsymmetricKeyPackage with symmetric encryption, where the sender and the receiver already share the necessary encryption key.

- EnvelopedData can be used to encrypt the AsymmetricKeyPackage with symmetric encryption, where the sender and the receiver do not share the necessary encryption key.

- AuthenticatedData can be used to protect the AsymmetricKeyPackage with message authentication codes, where key management information is handled in a manner similar to EnvelopedData.

- AuthEnvelopedData can be used to protect the AsymmetricKeyPackage with algorithms that support authenticated encryption, where key management information is handled in a manner similar to EnvelopedData.

5. Other Private-Key Format Considerations

This document defines the syntax and the semantics for a content type that exchanges asymmetric private keys. There are two other formats that have been used for the transport of asymmetric private keys:

- Personal Information Exchange (PFX) Syntax Standard [P12], which is more commonly referred to as PKCS #12 or simply P12, is a transfer syntax for personal identity information, including private keys, certificates, miscellaneous secrets, and extensions. OneAsymmetricKey, PrivateKeyInfo, and EncryptedPrivateKeyInfo can be carried in a P12 message. The private key information, OneAsymmetricKey and PrivateKeyInfo, are carried in the P12 keyBag BAG-TYPE. EncryptedPrivateKeyInfo is carried in the P12 pkcs8ShroudedKeyBag BAG-TYPE. In current implementations, the file extensions .pfx and .p12 can be used interchangeably.

- Microsoft's private-key proprietary transfer syntax. The .pvk file extension is used for local storage.

The .pvk and .p12/.pfx formats are not interchangeable; however, conversion tools exist to convert from one format to another.

To extract the private-key information from the AsymmetricKeyPackage, the encapsulating layers need to be removed. At a minimum, the outer ContentInfo [RFC5652] layer needs to be removed. If the AsymmetricKeyPackage is encapsulated in a SignedData [RFC5652], then the SignedData and EncapsulatedContentInfo layers [RFC5652] also need to be removed. The same is true for EnvelopedData, EncryptedData, and AuthenticatedData all from [RFC5652] as well as AuthEnvelopedData from [RFC5083]. Once all the outer layers are removed, there are as many sets of private-key information as there are OneAsymmetricKey structures. OneAsymmetricKey and PrivateKeyInfo are the same structure; therefore, either can be saved as a .p8 file or copied in to the P12 KeyBag BAG-TYPE. Removing encapsulating security layers will invalidate any signature and may expose the key to unauthorized disclosure.

.p8 files are sometimes PEM-encoded. When .p8 files are PEM encoded they use the .pem file extension. PEM encoding is either the Base64 encoding, from Section 4 of [RFC4648], of the DER-encoded EncryptedPrivateKeyInfo sandwiched between:

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
-----END ENCRYPTED PRIVATE KEY-----
```

or the Base64 encoding, see Section 4 of [RFC4648], of the DER-encoded PrivateKeyInfo sandwiched between:

```
-----BEGIN PRIVATE KEY-----
-----END PRIVATE KEY-----
```

6. Security Considerations

Protection of the private-key information is vital to public-key cryptography. Disclosure of the private-key material to another entity can lead to masquerades. The encryption algorithm used in the encryption process must be as 'strong' as the key it is protecting.

The asymmetric key package contents are not protected. This content type can be combined with a security protocol to protect the contents of the package.

7. IANA Considerations

This document makes use of object identifiers to identify a CMS content type and the ASN.1 module found in Appendix A. The CMS content type OID is registered in a DoD arc. The ASN.1 module OID is registered in an arc delegated by RSADSI to the SMIME Working Group. No further action by IANA is necessary for this document or any anticipated updates.

This specification also defines a new media subtype that IANA has registered at http://www.iana.org/.

7.1. Registration of media subtype application/pkcs8

Type name: application

Subtype name: pkcs8

Required parameters: None

Optional parameters: None

Encoding considerations: binary

Security considerations: Carries a cryptographic private key.
See section 6.

Interoperability considerations:

The PKCS #8 object inside this media type MUST be DER-encoded PrivateKeyInfo.

Published specification: RFC 5958

Applications which use this media type:

Any MIME-compliant transport that processes asymmetric keys.

Additional information:

Magic number(s): None
File extension(s): .p8
Macintosh File Type Code(s):

Person & email address to contact for further information:

Sean Turner <turners@ieca.com>

Restrictions on usage: none

Author:

Sean Turner <turners@ieca.com>

Intended usage: COMMON

Change controller:

The IESG

8. References

8.1. Normative References

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

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

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

[RFC5911] Hoffman, P. and J. Schaad, "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911, June 2010.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, June 2010.

[X.680] ITU-T Recommendation X.680 (2002) | ISO/IEC 8824-1:2002. Information Technology - Abstract Syntax Notation One.

[X.681] ITU-T Recommendation X.681 (2002) | ISO/IEC 8824-2:2002. Information Technology - Abstract Syntax Notation One: Information Object Specification.

[X.682] ITU-T Recommendation X.682 (2002) | ISO/IEC 8824-3:2002. Information Technology - Abstract Syntax Notation One: Constraint Specification.

[X.683] ITU-T Recommendation X.683 (2002) | ISO/IEC 8824-4:2002. Information Technology - Abstract Syntax Notation One: Parameterization of ASN.1 Specifications.

[X.690] ITU-T Recommendation X.690 (2002) | ISO/IEC 8825-1:2002. Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).

8.2. Informative References

[P12] RSA Laboratories, "PKCS #12 v1.0: Personal Information Exchange Syntax", June 1999.

[RFC2985] Nystrom, M. and B. Kaliski, "PKCS #9: Selected Object Classes and Attribute Types Version 2.0", RFC 2985, November 2000.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC5083] Housley, R., "Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type", RFC 5083, November 2007.

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

[X.520] ITU-T Recommendation X.520 (2005) | ISO/IEC 9594-6:2005, Information technology - Open Systems Interconnection - The Directory: Selected attribute types.

[RFC5915] Turner, S. and D. Brown, "Elliptic Curve Private Key Structure", RFC 5915, June 2010.

Appendix A. ASN.1 Module

This annex provides the normative ASN.1 definitions for the structures described in this specification using ASN.1 as defined in [X.680] through [X.683].

```
AsymmetricKeyPackageModuleV1
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) modules(0) id-mod-asymmetricKeyPkgV1(50) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL

IMPORTS

-- FROM New SMIME ASN.1 [RFC5911]

Attribute{}, CONTENT-TYPE
 FROM CryptographicMessageSyntax-2009
   { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
     smime(16) modules(0) id-mod-cms-2004-02(41) }

-- From New PKIX ASN.1 [RFC5912]
ATTRIBUTE
 FROM PKIX-CommonTypes-2009
   { iso(1) identified-organization(3) dod(6) internet(1)
     security(5) mechanisms(5) pkix(7) id-mod(0)
     id-mod-pkixCommon-02(57) }

-- From New PKIX ASN.1 [RFC5912]

AlgorithmIdentifier{}, ALGORITHM, PUBLIC-KEY, CONTENT-ENCRYPTION
  FROM AlgorithmInformation-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-algorithmInformation-02(58) }

;

ContentSet CONTENT-TYPE ::= {
 ct-asymmetric-key-package,
 ... -- Expect additional content types --
}
```

```
ct-asymmetric-key-package CONTENT-TYPE ::=
 { AsymmetricKeyPackage IDENTIFIED BY id-ct-KP-aKeyPackage }

id-ct-KP-aKeyPackage OBJECT IDENTIFIER ::=
  { joint-iso-itu-t(2) country(16) us(840) organization(1)
      gov(101) dod(2) infosec(1) formats(2)
      key-package-content-types(78) 5
  }

AsymmetricKeyPackage ::= SEQUENCE SIZE (1..MAX) OF OneAsymmetricKey

OneAsymmetricKey ::= SEQUENCE {
  version                   Version,
  privateKeyAlgorithm       PrivateKeyAlgorithmIdentifier,
  privateKey                PrivateKey,
  attributes            [0] Attributes OPTIONAL,
  ...,
  [[2: publicKey        [1] PublicKey OPTIONAL ]],
  ...
}

PrivateKeyInfo ::= OneAsymmetricKey

-- PrivateKeyInfo is used by [P12]. If any items tagged as version
-- 2 are used, the version must be v2, else the version should be
-- v1. When v1, PrivateKeyInfo is the same as it was in [RFC5208].

Version ::= INTEGER { v1(0), v2(1) } (v1, ..., v2)

PrivateKeyAlgorithmIdentifier ::= AlgorithmIdentifier
                                    { PUBLIC-KEY,
                                      { PrivateKeyAlgorithms } }

PrivateKey ::= OCTET STRING
                  -- Content varies based on type of key. The
                  -- algorithm identifier dictates the format of
                  -- the key.

PublicKey ::= BIT STRING
                  -- Content varies based on type of key. The
                  -- algorithm identifier dictates the format of
                  -- the key.

Attributes ::= SET OF Attribute { { OneAsymmetricKeyAttributes } }

OneAsymmetricKeyAttributes ATTRIBUTE ::= {
  ... -- For local profiles
}
```

```
   -- An alternate representation that makes full use of ASN.1
   -- constraints follows.  Also note that PUBLIC-KEY needs to be
   -- imported from the new PKIX ASN.1 Algorithm Information module
   -- and PrivateKeyAlgorithms needs to be commented out.

   -- OneAsymmetricKey ::= SEQUENCE {
   --  version                   Version,
   --  privateKeyAlgorithm       SEQUENCE {
   --   algorithm                  PUBLIC-KEY.&id({PublicKeySet}),
   --    parameters                PUBLIC-KEY.&Params({PublicKeySet}
   --                                {@privateKeyAlgorithm.algorithm})
   --                              OPTIONAL}
   --  privateKey                OCTET STRING (CONTAINING
   --                              PUBLIC-KEY.&PrivateKey({PublicKeySet}
   --                              {@privateKeyAlgorithm.algorithm})),
   --  attributes           [0] Attributes OPTIONAL,
   --  ...,
   --  [[2: publicKey       [1] BIT STRING (CONTAINING
   --                              PUBLIC-KEY.&Params({PublicKeySet}
   --                              {@privateKeyAlgorithm.algorithm})
   --                              OPTIONAL,
   --  ...
   --    }

   EncryptedPrivateKeyInfo ::= SEQUENCE {
     encryptionAlgorithm  EncryptionAlgorithmIdentifier,
     encryptedData        EncryptedData }

   EncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
                                       { CONTENT-ENCRYPTION,
                                         { KeyEncryptionAlgorithms } }

   EncryptedData ::= OCTET STRING -- Encrypted PrivateKeyInfo

   PrivateKeyAlgorithms ALGORITHM ::= {
     ... -- Extensible
   }

   KeyEncryptionAlgorithms ALGORITHM ::= {
     ... -- Extensible
   }

   END
```

Acknowledgements

Many thanks go out to the Burt Kaliski and Jim Randall at RSA. Without the prior version of the document, this one wouldn't exist.

I'd also like to thank Pasi Eronen, Roni Even, Alfred Hoenes, Russ Housley, Jim Schaad, and Carl Wallace.

Author's Address

Sean Turner
IECA, Inc.
3057 Nutley Street, Suite 106
Fairfax, VA 22031
USA

EMail: turners@ieca.com